

1

The Decentralized Web and the Cerebral Cortex

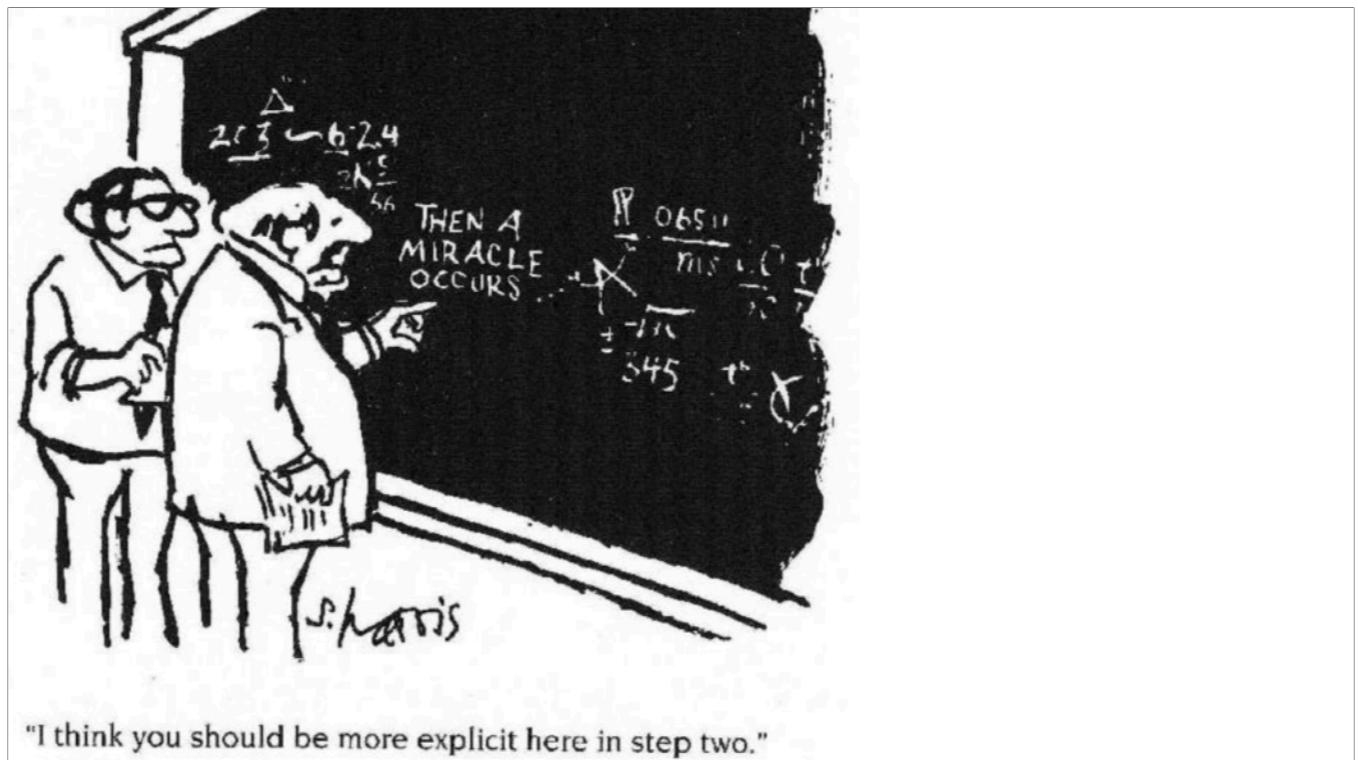
Shared Challenges, Shared Solutions

David Strayhorn MD, PhD

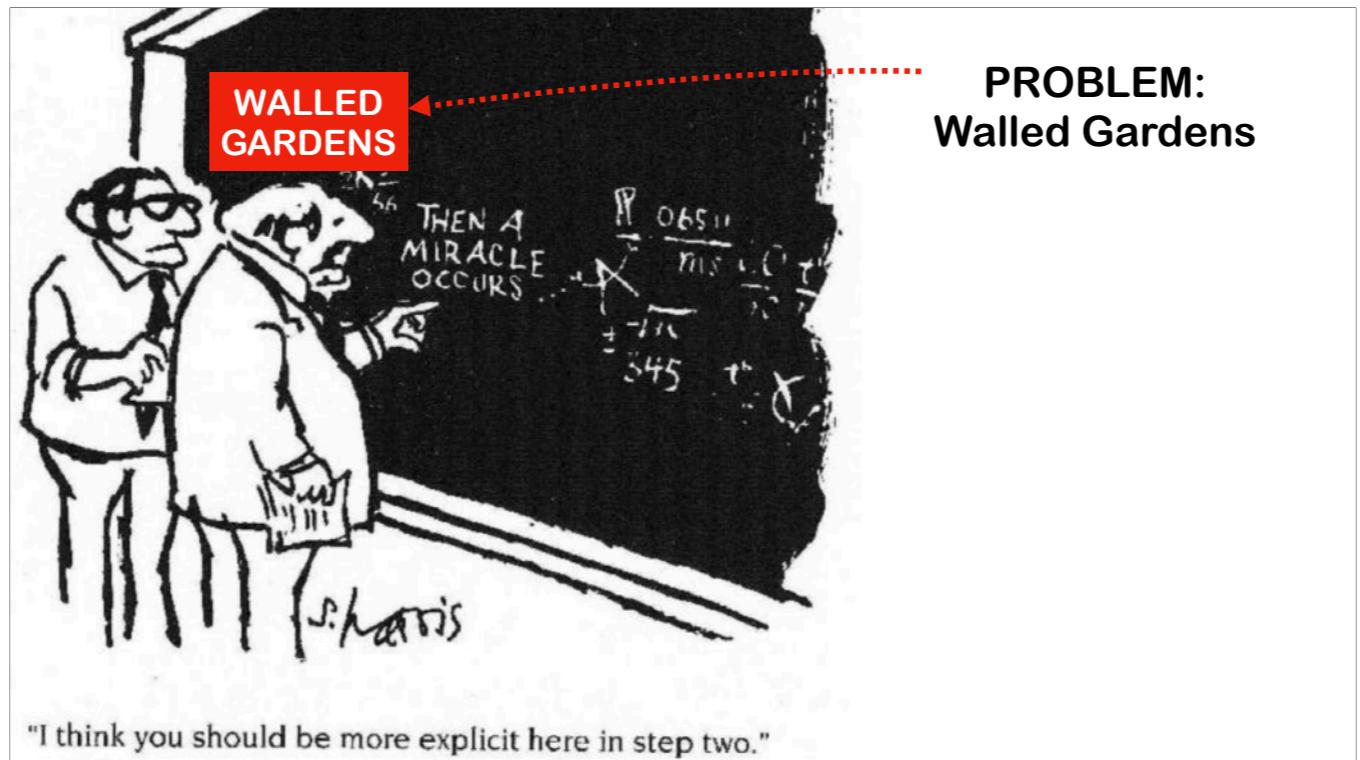
~ 2 hour version

[start 01]

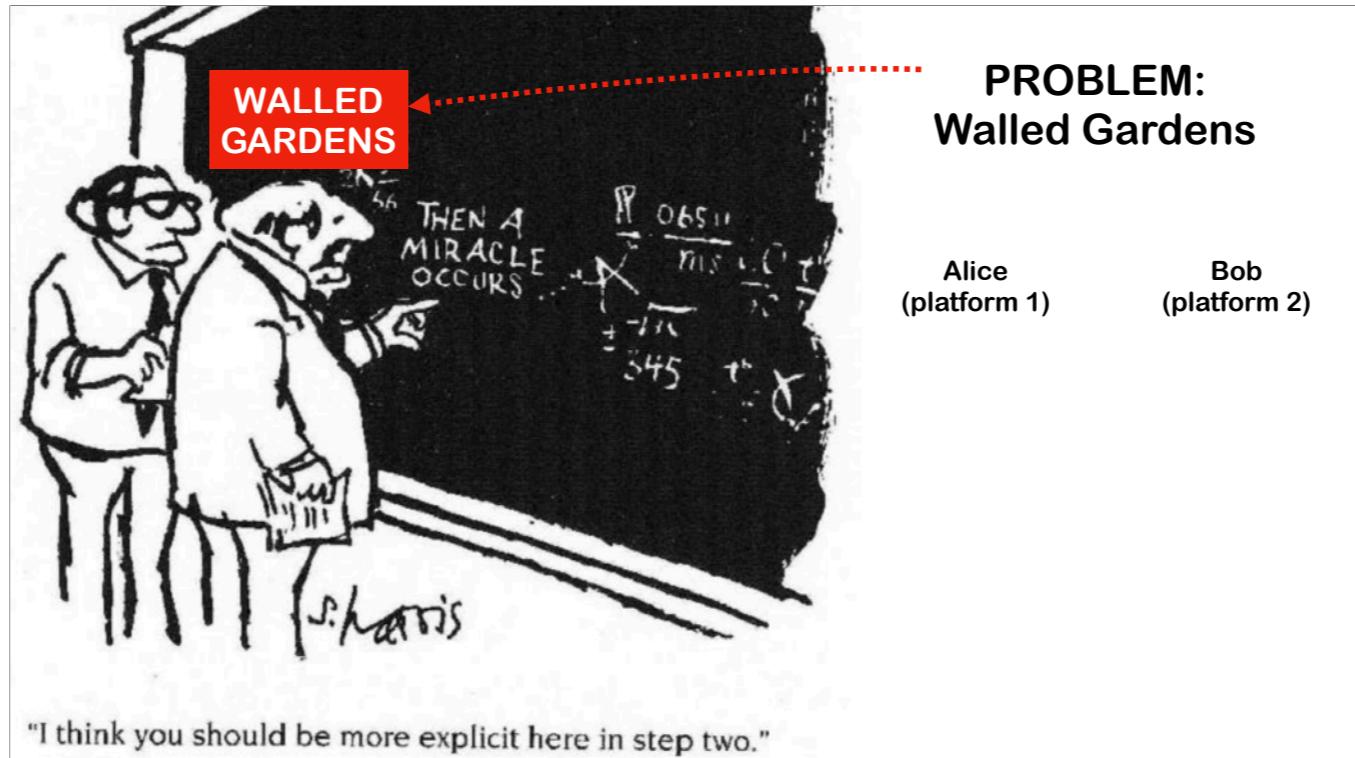
Hello everyone. In this video I am going to be discussing two systems — the decentralized web and the cerebral cortex — that are obviously different in many ways; but which i postulate may share some surprising similarities in terms of the challenges that they face when it comes to organization of data, and in terms of the solutions to those problems that either should be utilized or are potentially already being utilized. Primarily I will be talking about the decentralized web, about the problem of Walled Gardens, and about a potential solution to that problem; but I will offer the conjecture that a quite similar set of challenges, and quite similar method of solving them, are already characteristic of the cerebral cortex.



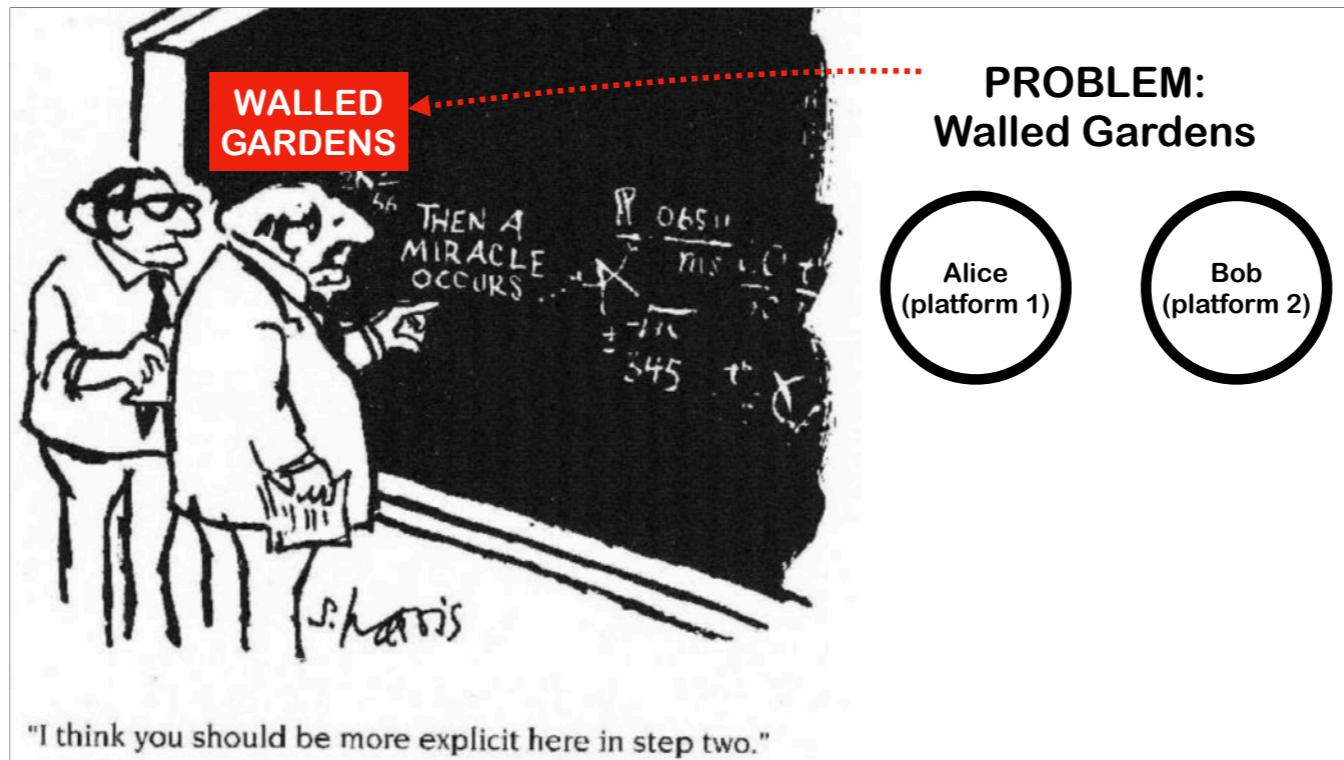
Many of you have probably seen this Sidney Harris cartoon: a mathematician is trying to solve a particularly thorny problem, and for his solution he says “then a miracle occurs.” Well, the primary challenge I’ll be talking about



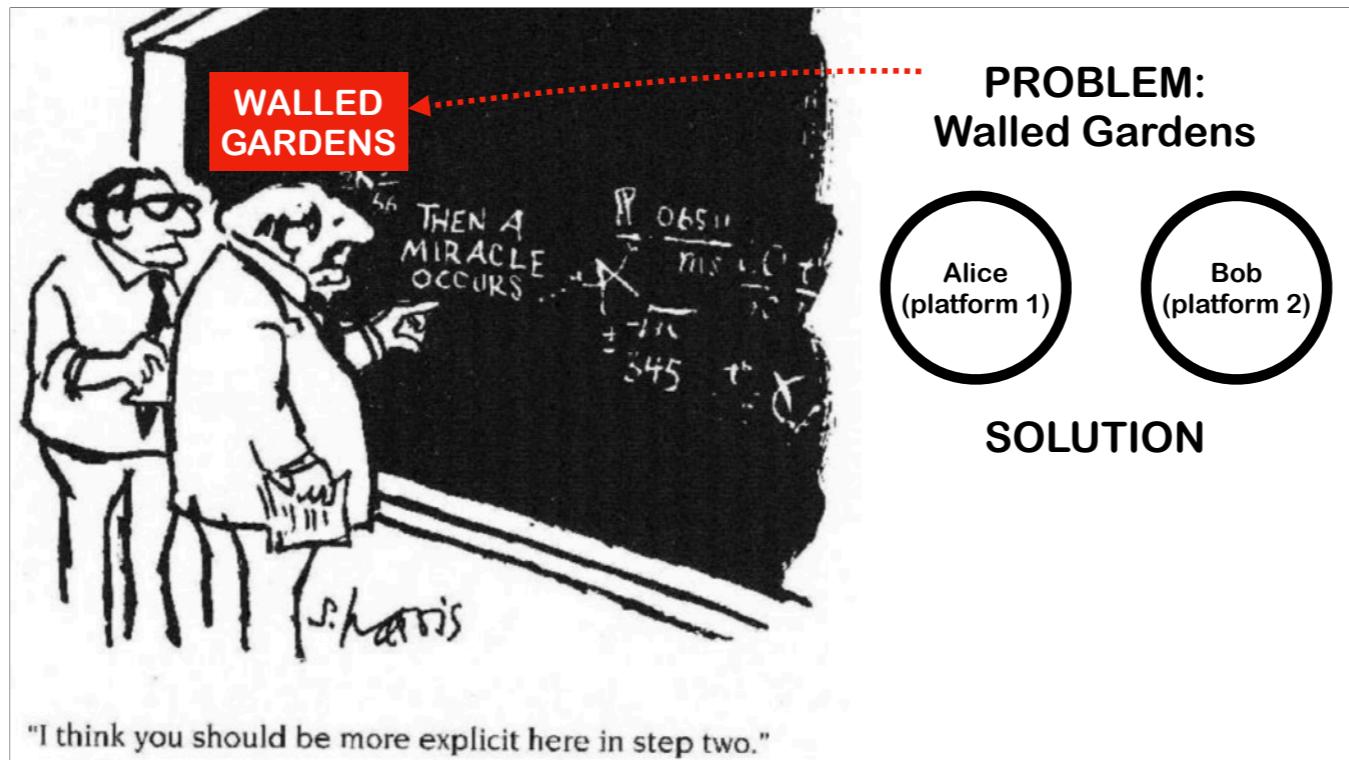
is the problem of Walled Gardens. Consider two users



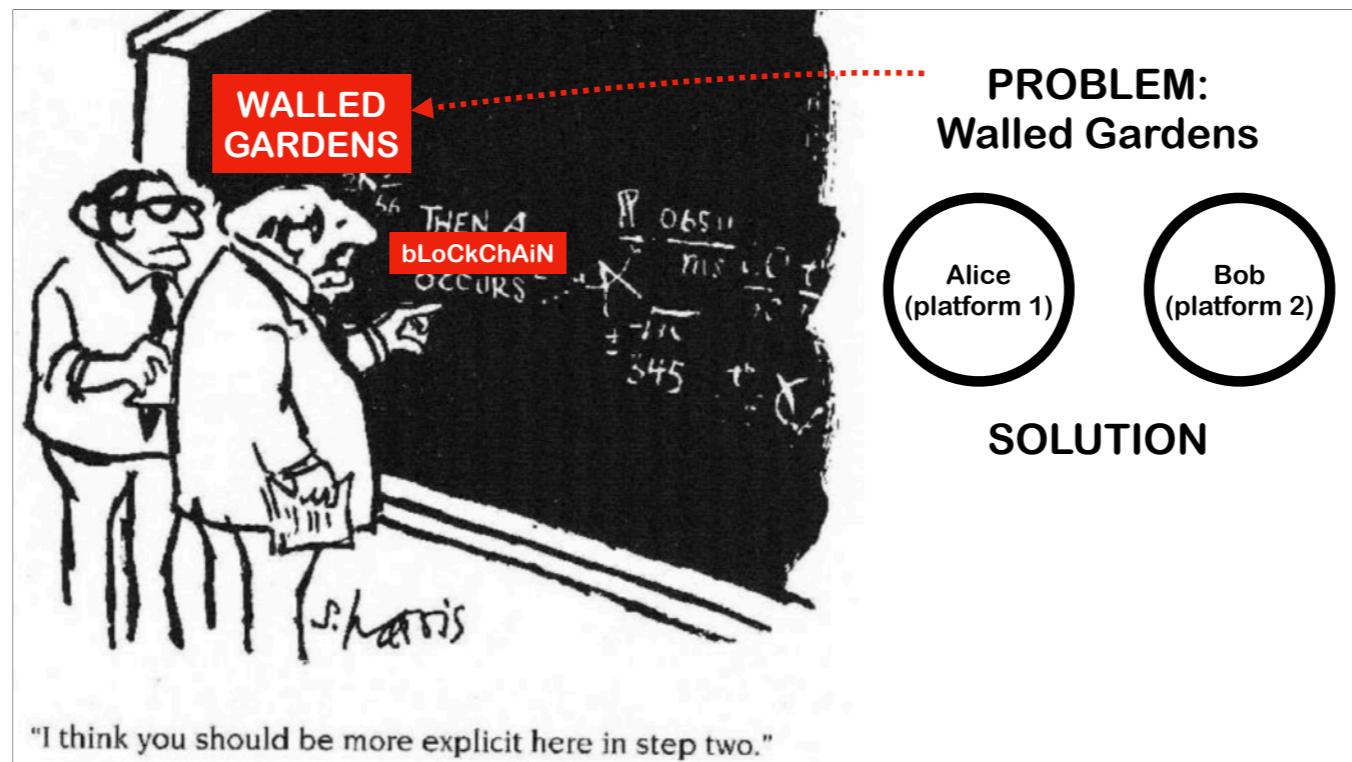
Alice and Bob, who have user accounts on 2 distinct platforms. Platforms as we know them today — platforms like LinkedIn, eBay, Facebook, Twitter, Reddit, and so on — are designed to facilitate communication and to facilitate data portability within the platform, but not between platforms. I'm going to draw a circle around each of them



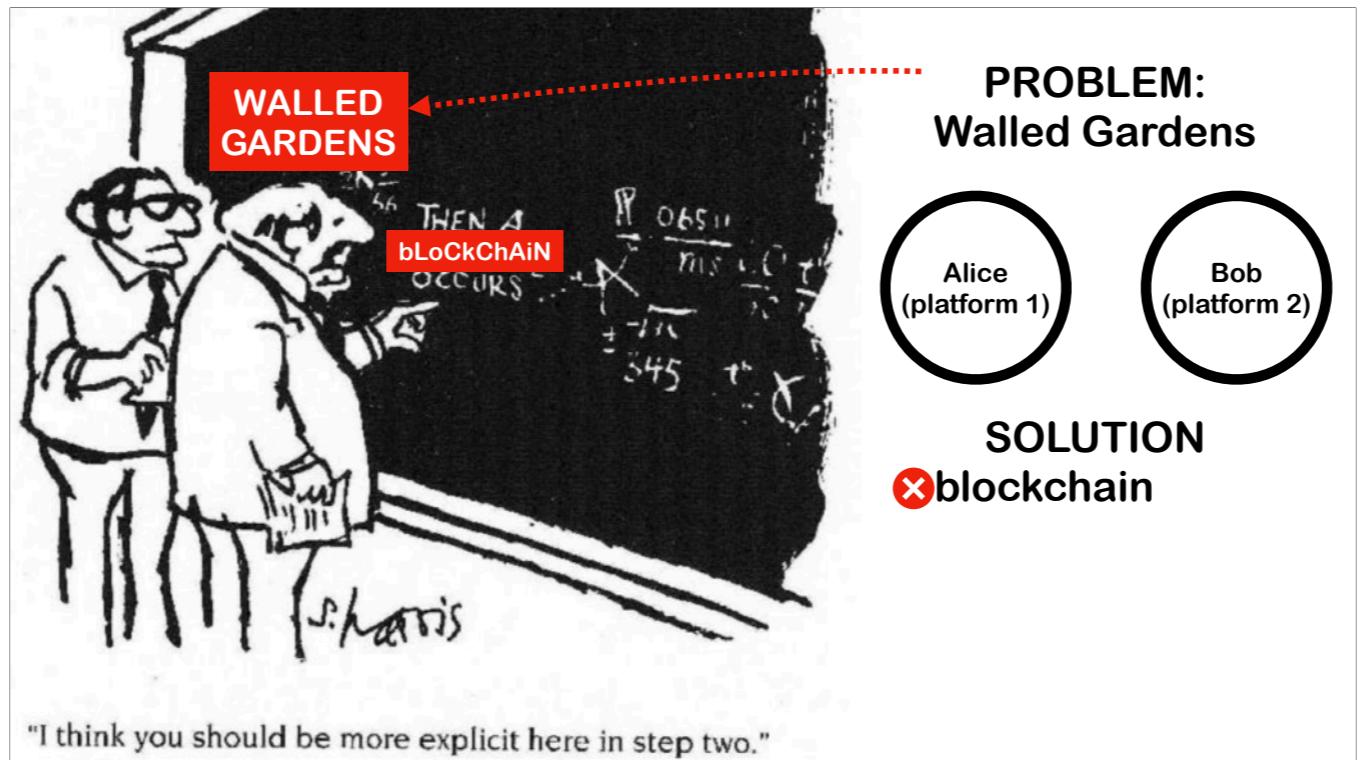
like so, to represent these barriers to communication, these barriers to data portability, that exist between Alice and Bob by virtue of the fact that they are on separate platforms. This is the problem of walled gardens. Why do these barriers exist? There may be different ways of conceptualizing this problem; but in this video I'll be focusing on the fact that they do not share a common language. More specifically: they generally speaking don't format data in the same way.



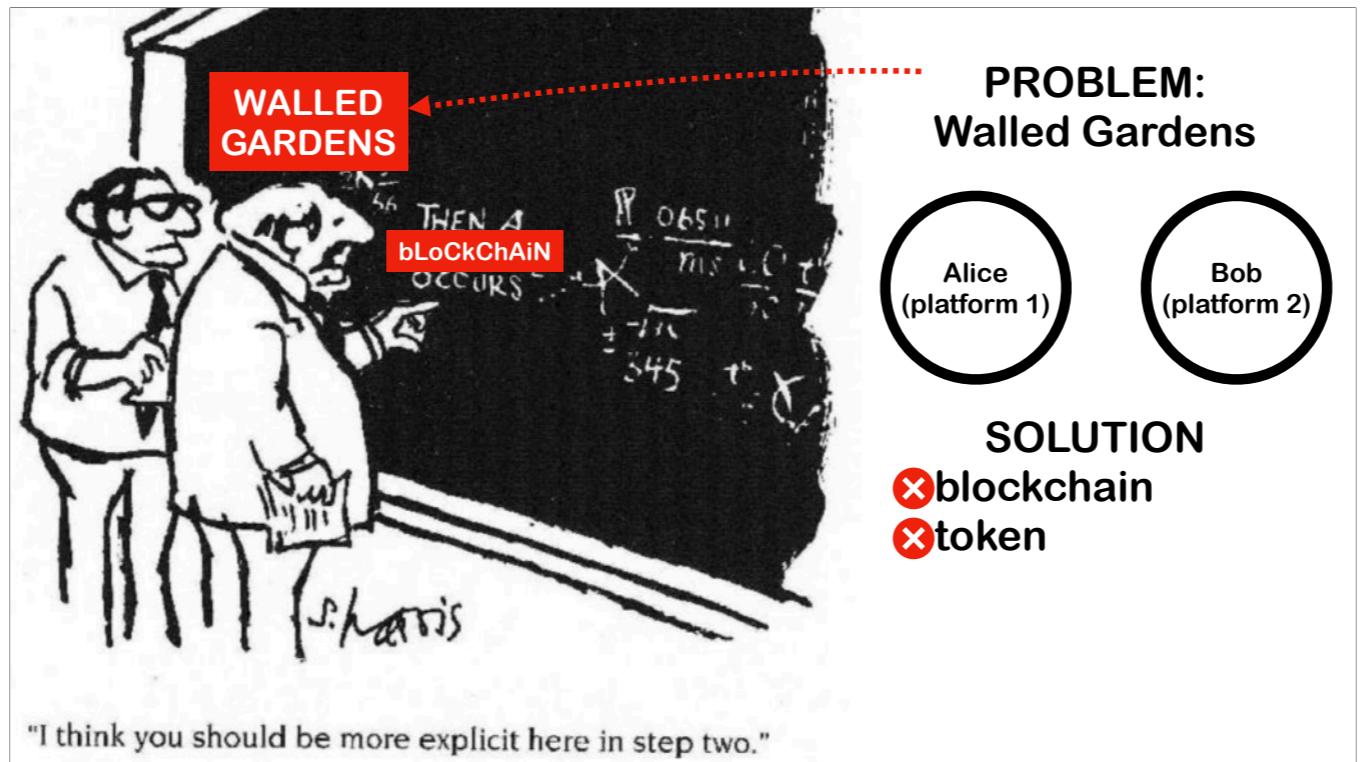
Now, when it comes to proposing solutions, in the cartoon the mathematician says: Then a miracle occurs. In the case of decentralized web, this is where many people might say:



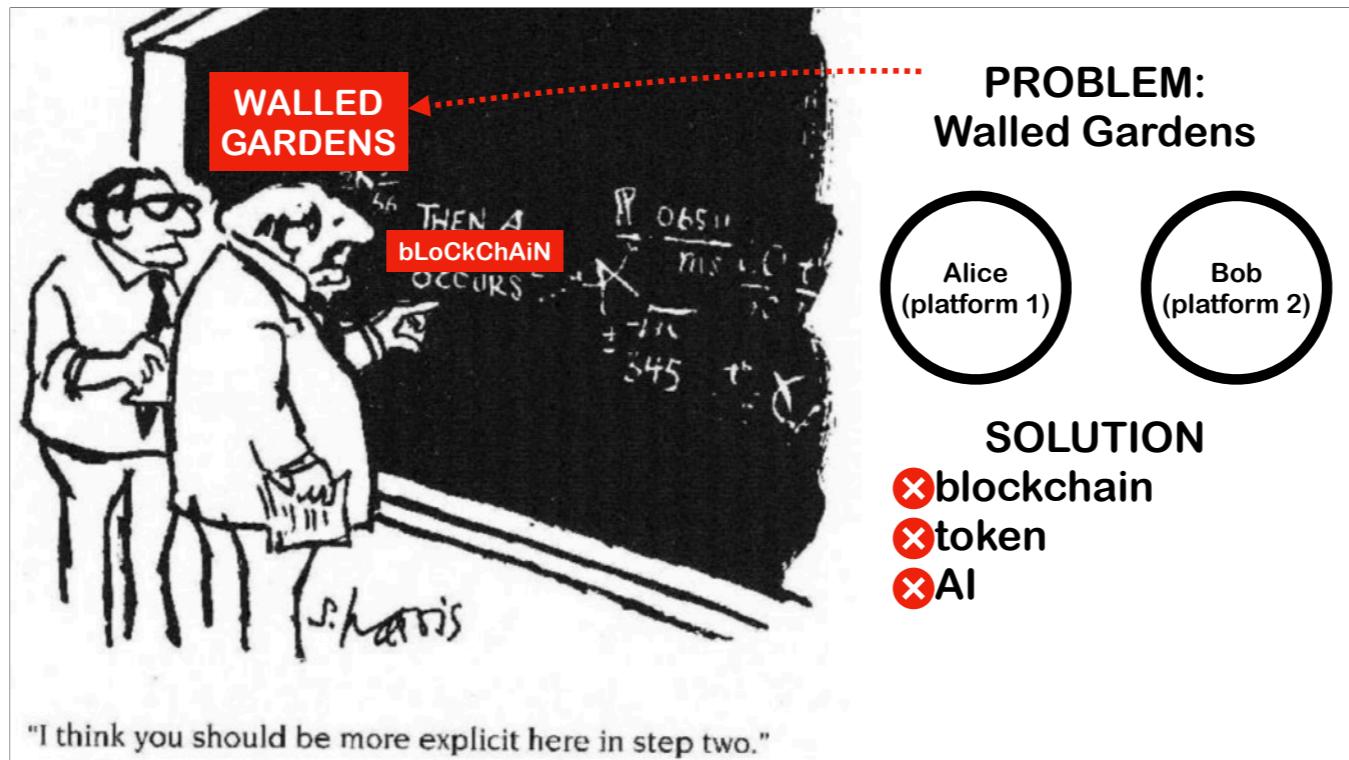
Then a blockchain occurs. So let me make it clear, up front:



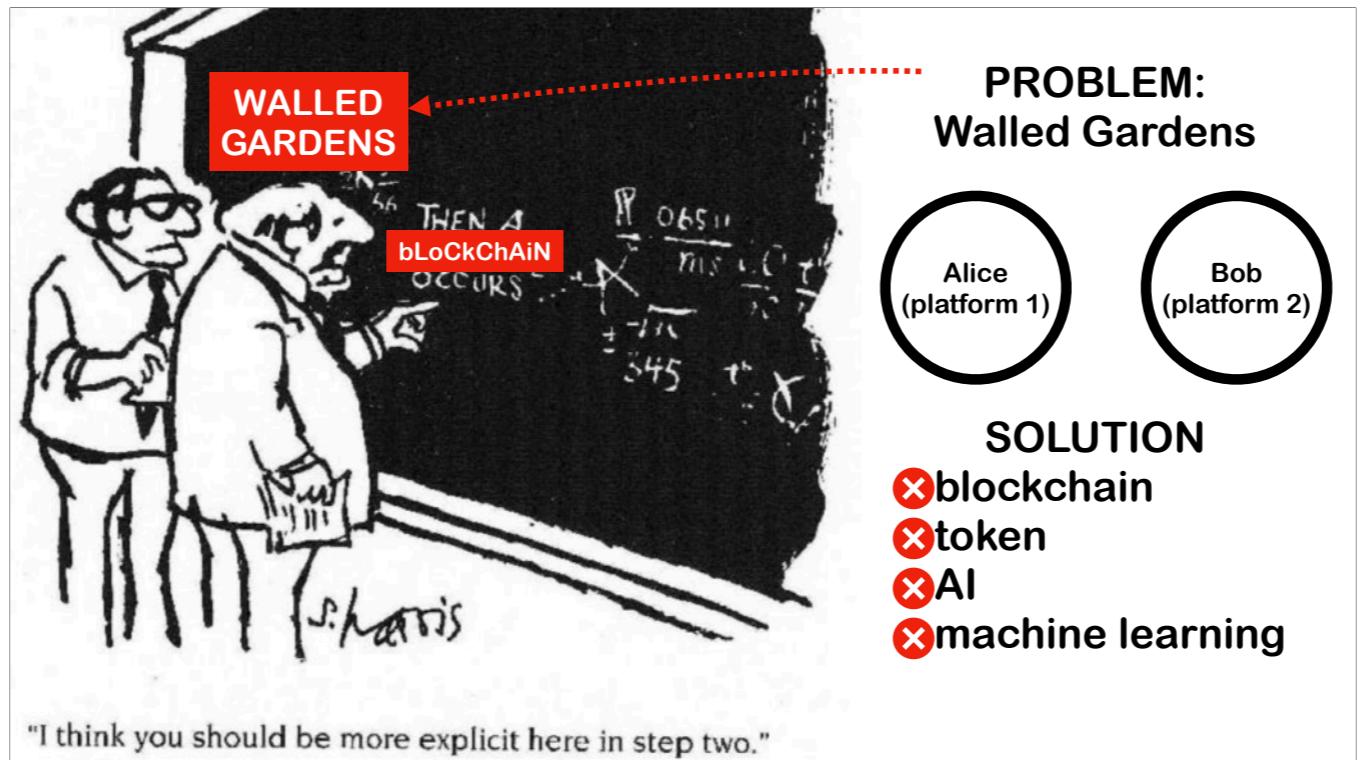
I will not be proposing a blockchain. I won't be telling you "then a blockchain occurs."



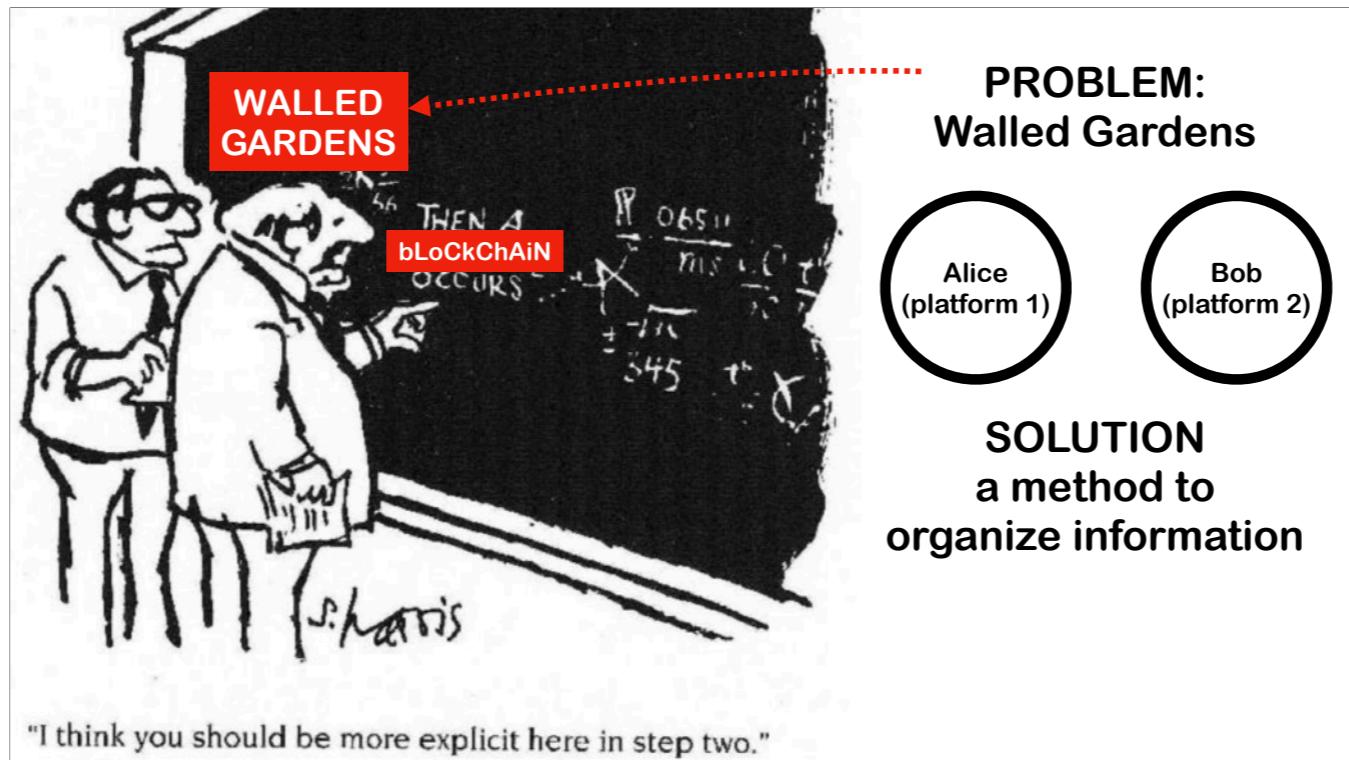
I will not be promoting a token,



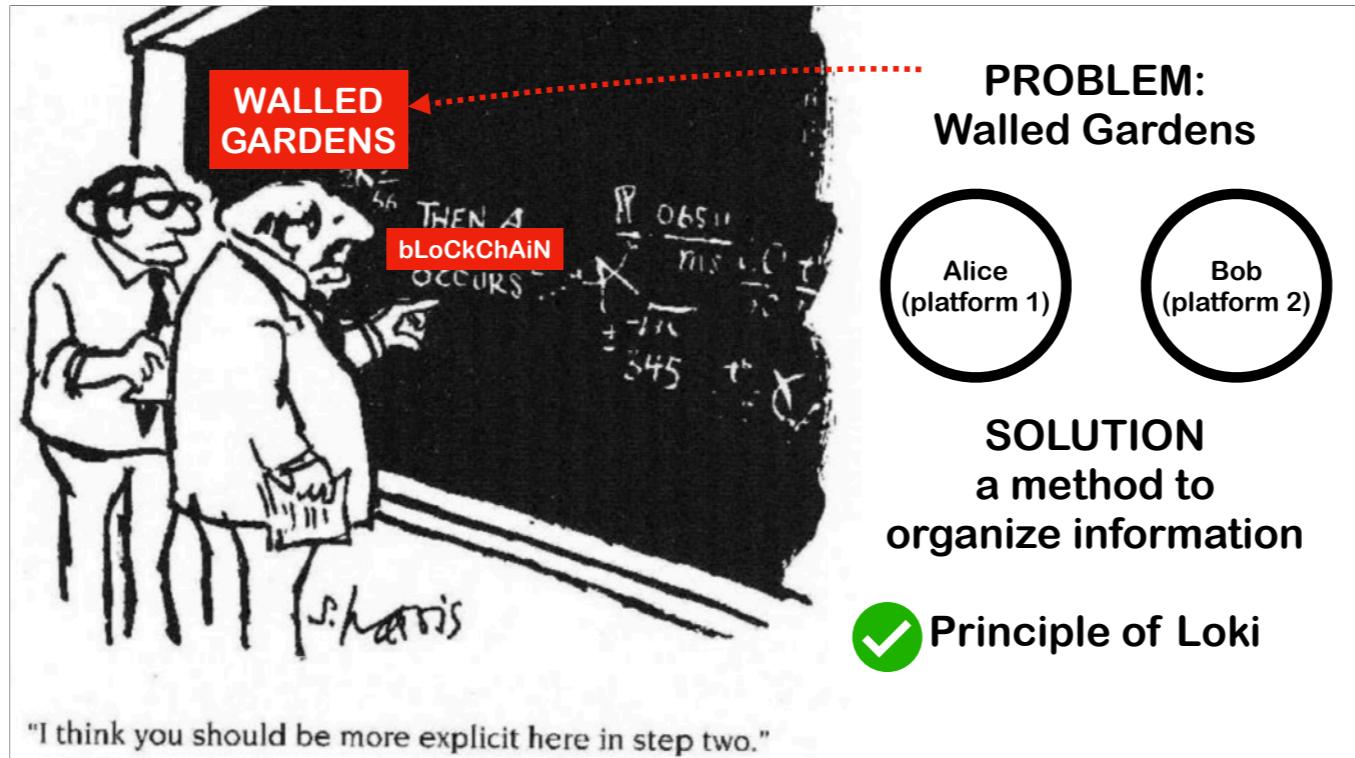
I won't be proposing that we need artificial intelligence,



or machine learning, or any other black box miracle solution.



The solution I will be proposing in this talk is an IDEA, in fact, it's a pretty simple one. This IDEA provides the foundation for a method to organize information. And at the risk of sounding corny, I am going to give this idea a name, and call it



the Principle of Loki. Now, I didn't choose that name to refer explicitly to Loki the brother of Thor and the god of mischief ...

Solution: Principle of Loki

Localization of Keys → “Loki”

I choose Loki as being shorthand for “localization of keys.”

[end 01]

2

Solution: Principle of Loki

Localization of Keys → “Loki”

[start 02]

The idea behind the Principle of Loki starts with the observation

Solution: Principle of Loki

Localization of Keys → “Loki”



data

that any given file — any given piece of data —

Solution: Principle of Loki

Localization of Keys → “Loki”



has a format, and without knowing the format, the file itself is for all practical purposes, meaningless. Useless. A record of the file's format is, in a manner of speaking, a key to the file. Without having this key, or knowing the format, and — most importantly — without having a shared consensus with other users on this format, you can't read a file, you can't write a file, you can't even request other users to send you a file of files bc they (and by they I mean their software) — they won't know what you're talking about, because you don't have the proper tools to communicate what kind of file it is that you're even requesting.

Solution: Principle of Loki

Localization of Keys → “Loki”



format



data

That format needs to be recorded explicitly, SOMEWHERE.

Solution: Principle of Loki

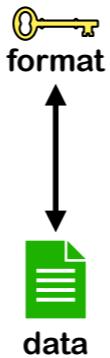
For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



The principle states that within any self contained piece of software, like an app, for any piece of data, there must always be a simple, straightforward way to connect the data with a specification of its format. And I don't mean that you bury it in documentation. If you can locate one, it should be a simple matter to locate the other. It should even be possible to automate that process of localization.

Solution: Principle of Loki

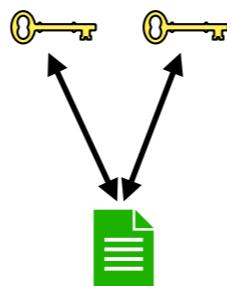
For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



It's a pretty straightforward idea. The entire purpose of my talk is to explain what I mean by this principle, how I implement it, and why I believe it is necessary for us to incorporate this idea into the fabric of the dWeb, if it is ever to live up to the full potential that so many of us believe it to have.

Solution: Principle of Loki

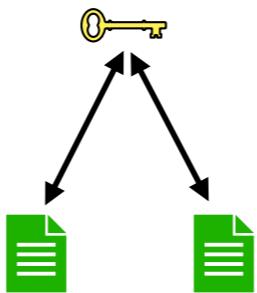
For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



As part of this Principle, specification of any given file format can be broken up into pieces; it doesn't always have to be in one place.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



On the other hand, multiple files can share the same format. And typically this is going to be the case. Imagine files of user data: multiple users, multiple files, but they all may use the exact same format.

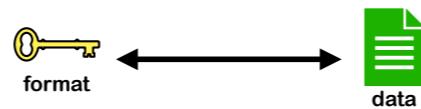
Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



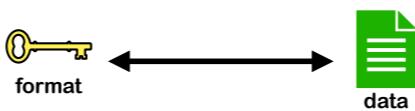
Implementation:



This Principle is the starting point for me to begin building a pair of apps that I am calling the CG and the Grapevine, two apps that I will be presenting in this video. They each can function independently, but each relies upon the other to realize its full potential, and each is designed with this pairing in mind.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



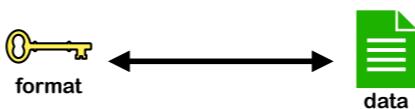
Implementation:



I will be describing them in considerable detail in this video. If you are a developer who might be thinking of working on these tools, or using them for your own projects, you'll want to follow these details closely. If you're not a prospective developer, if you're just trying to get the big picture, you don't have to follow the details closely.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



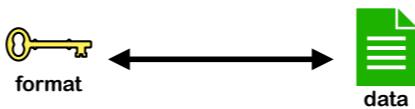
Implementation:



The reason I present them is that I want you to appreciate the fact that these details are the natural and inevitable consequence of this one simple starting point, which is the Principle of Loki. The Principle can be encapsulated in just one sentence. Once you decide to accept the Principle and take it seriously, and once you make some initial decisions like using JSON files to store data which is what I'm using, the full designs of both the Concept Graph and the Grapevine take care of themselves; they fall into place.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



Implementation:

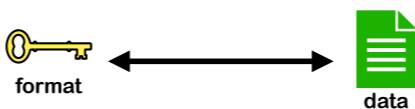


LOOSE CONSENSUS

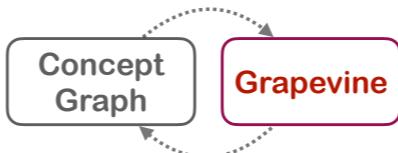
These two apps will be used within a dSystem to build what I call “loose consensus.” That’s one of the very important concepts in this video, and it means:

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



Implementation:



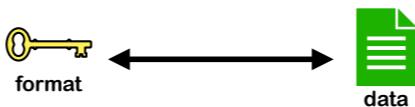
LOOSE CONSENSUS over questions (like how to format data!) that are

- mundane, tedious
- not controversial
- no Schelling point

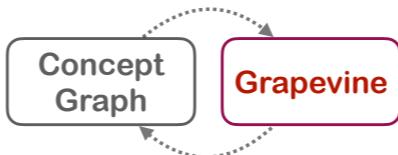
consensus on questions, like how to format data, questions that may be necessary for the sake of communication, but Q's that may be mundane, tedious, involve lots of details ... iow, boring! not particularly controversial, but Q's that may have a multitude of perfectly reasonable answers, around which no well defined Schelling point exists. On questions like this, everyone WANTS consensus, they just don't have tools to establish it. That is — for better or for worse — one of the essential features of decentralization.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



Implementation:



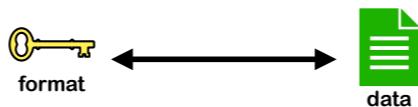
LOOSE CONSENSUS over questions (like how to format data!) that are

- mundane, tedious
- not controversial
- no Schelling point

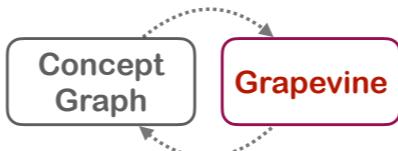
This is NOT a tool to establish consensus on CONTROVERSIAL questions. If people are enemies, this set of tools will not get them to agree on anything that is sufficiently controversial. This is one of the things that makes Loose Consensus distinct from something like Nakamoto consensus. With Nakamoto consensus, you're agreeing on a protocol, and it's all or nothing. With loose consensus, it's not all or nothing. You can agree on some elements, disagree on others. Like a language: we can agree on syntax, definitions of some words, but disagree on meanings of some other words, that's OK; small disagreements over meaning of one word doesn't COMPLETELY derail the English language or the ability to use it to communicate. In fact, if you're learning a new language, a tiny vocabulary, compared to nothing at all, goes quite a long way. loose consensus will be the same.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.

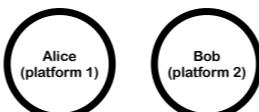


Implementation:



LOOSE CONSENSUS over questions (like how to format data!) that are

- mundane, tedious
- not controversial
- no Schelling point

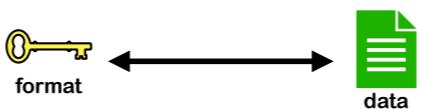


Without consensus on how to format data, platforms, whether they are proprietary or open source, will continue to be walled gardens.

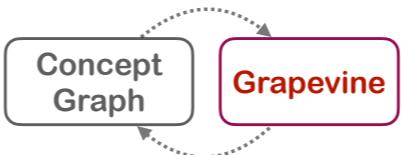
Loose consensus, built using the CG and the Grapevine, is what I propose to be a powerful method

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



Implementation:



LOOSE CONSENSUS over questions (like how to format data!) that are

- mundane, tedious
- not controversial
- no Schelling point

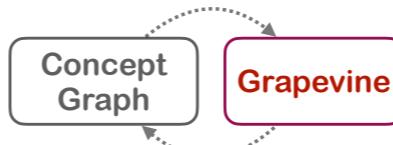
Alice
(platform 1) Bob
(platform 2)

to tear down these walls so that Alice and Bob, regardless of whether they are or are not on the same platform, regardless of how you even define platform, will find communication to be as easy and effective as if they were on the same platform.

[end 02]

3

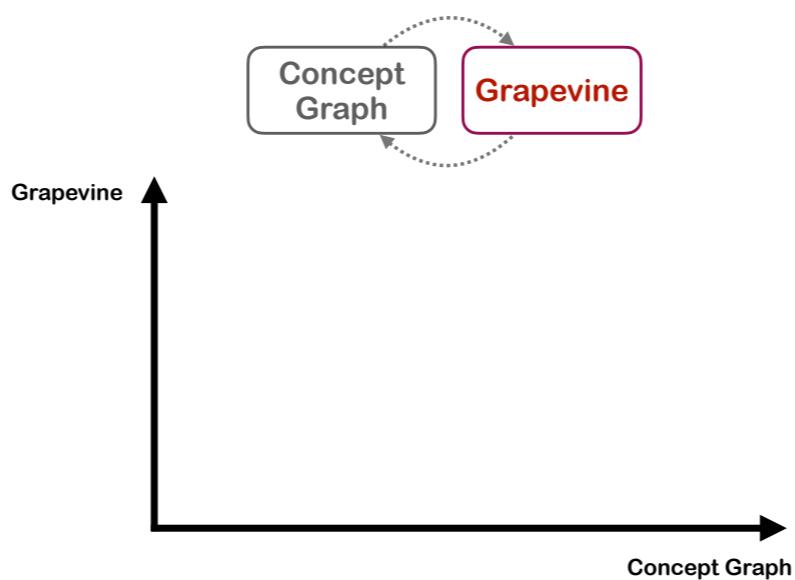
Solution: Principle of Loki



[start 03]

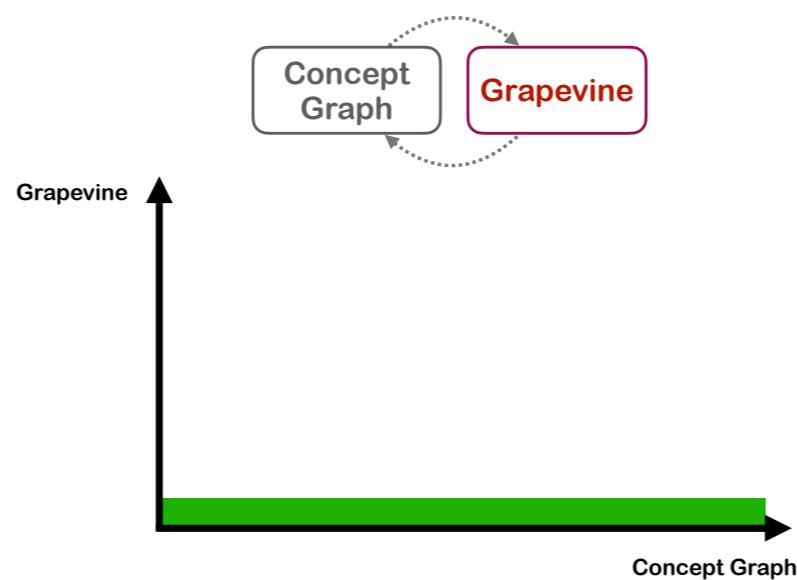
The CG and the Grapevine will be built (I should say are being built) as two, distinct, standalone apps. They each exist and can function independently; but each depends on the other in order to be maximally useful, and I hope that will be clear by the end of this talk.

Solution: Principle of Loki



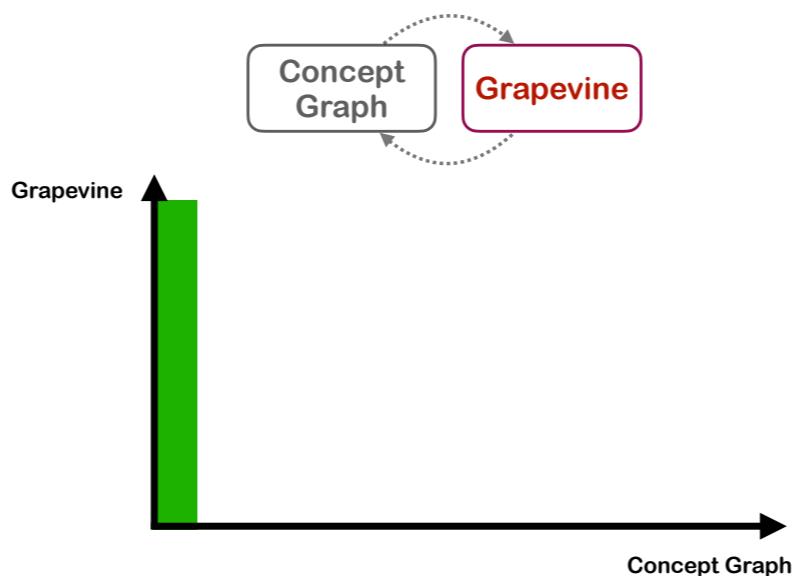
They're like orthogonal vectors.

Solution: Principle of Loki



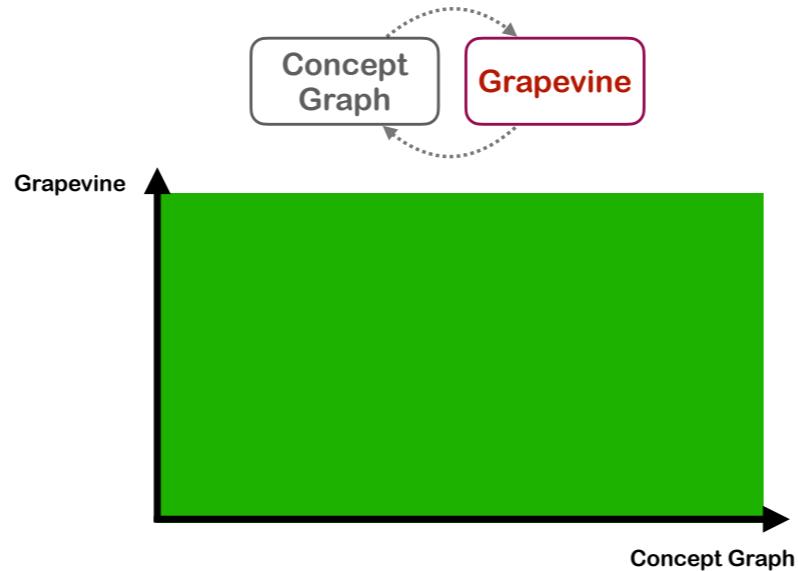
Length is good.

Solution: Principle of Loki



Height is good.

Solution: Principle of Loki

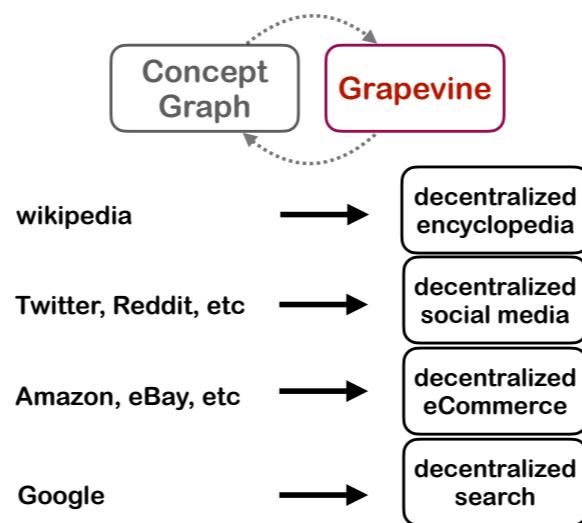


But most powerful when used together in combination.

Solution: Principle of Loki

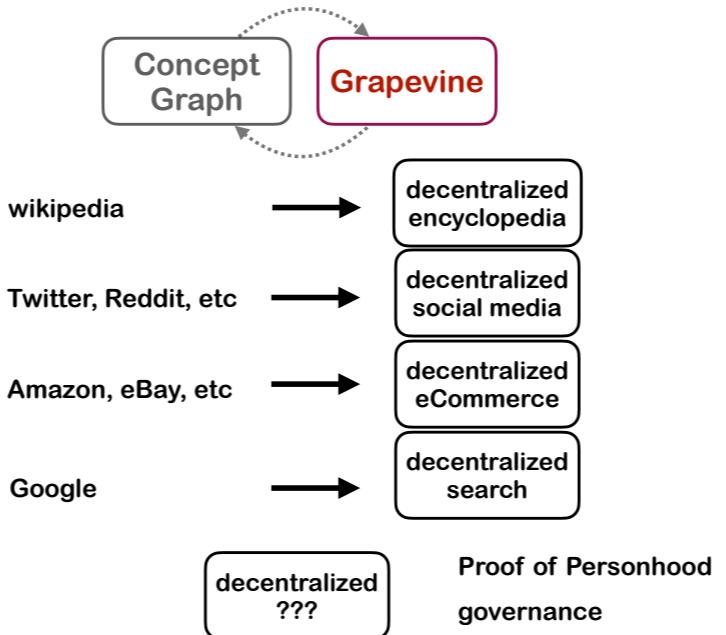


Solution: Principle of Loki



And once these two apps are done, they will form the basis for additional decentralized, open source apps which will allow us to replace — allow us to exit — from, and phase out of today's centralized proprietary platforms like Twitter, google, eBay and so on. IOW, these apps will allow us to build the decentralized Web. Without walled gardens.

Solution: Principle of Loki



At the risk of getting ahead of myself, I believe dWeb will allow us to build additional apps and tools to do things that we have never done before: decentralized PoP, decentralized governance, and so on.

Solution: Principle of Loki (PoL)

By the end of my talk I hope I will have convinced you that the Principle of Loki is worthwhile; and I want anyone watching this talk who is working on the decentralized web, no matter what solutions you envision, to ask yourself: do I value interoperability? and if so, should I be incorporating this principle (the PoL) into the tools I'm building. I hope to convey to you two ideas:

Solution: Principle of Loki (PoL)

I CLAIM:

**The PoL SHOULD BE built into the decentralized web
(although we haven't done it yet!)**

first of all, I claim that this Principle MUST be incorporated into the dWeb, if the dWeb is ever to reach its full potential. Now: we haven't done that yet! If we don't, I don't care how cool your project is or how excited you are about it: you are going to be forever frustrated at your lack of traction. Everyone who has tried to build some variation on the idea of WoT starting in 1991 when Phil Zimmermann released Pretty Good Privacy has failed to gain meaningful traction; the kind of traction that we all know is possible, but which has, so far, been elusive. As long as we fail to recognize and take seriously the Principle of Loki — I don't necessarily mean MY apps, but just the idea itself — we will continue to be frustrated in our efforts.

Solution: Principle of Loki (PoL)

I CLAIM:

**The PoL SHOULD BE built into the decentralized web
(although we haven't done it yet!)**

I CONJECTURE:

**The PoL IS ALREADY BUILT into the cerebral cortex
(although we haven't proven that yet!)**

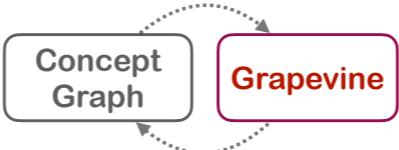
Second: I am going to conjecture that the PoL is ALREADY woven into the cerebral cortex. IOW nature is ahead of us. This second point is merely a conjecture. I'm not going to be presenting evidence for it; I merely am putting forth the idea. Why? bc I think it's a cool idea. and bc I think it might even be TRUE. But mostly I want to drive home the idea that the PoL — on the one hand: it is simple, straightforward to state; it can after all be put into a single sentence; — but on the other hand: once a few initial choices are made regarding data format, a lot of the details on how exactly to implement this idea fall naturally into place; and this can happen with a diverse array of applications spanning from computer science to the brain; wherever decentralization and walled gardens is a problem.

INTEROPERABILITY is the key



Regarding the two apps that I am describing in this talk, the CG and the Grapevine. These two apps represent my best attempt at an implementation of the PoL. But I'd like to point out that my implementation will not be the only possible implementation of this principle for the dWeb. I anticipate, indeed I count on the fact, that other people will build other implementations of the PoL. When I talk about interoperability, I am referring to the fact that I want users data to be portable, and I want various decentralized platforms to be interoperable; but more than that, I am referring to the fact that my implementation and someone else's implementation of the PoL must be interoperable. That's the goal.

INTEROPERABILITY is the key



... like BTC or LN wallets!

I think that in some ways, the situation I am in here with the CG and the Grapevine, is analogous to the situation that Jack Mallers is in with Strike. I say this based on some interviews and podcasts where I've heard him speak. He recognizes that all Lightning apps must be interoperable; and that includes Strike wallet with other lightning wallets. If they're not, he's failed, and lightning has failed. Same thing with the CG and the Grapevine. They ideally will be interoperable with whatever solutions other people may come up with. It's one reason I'm not trying to shoehorn in a token or a blockchain and building something around it. It would get in the way of achieving what I want to achieve.

[end 03]

4

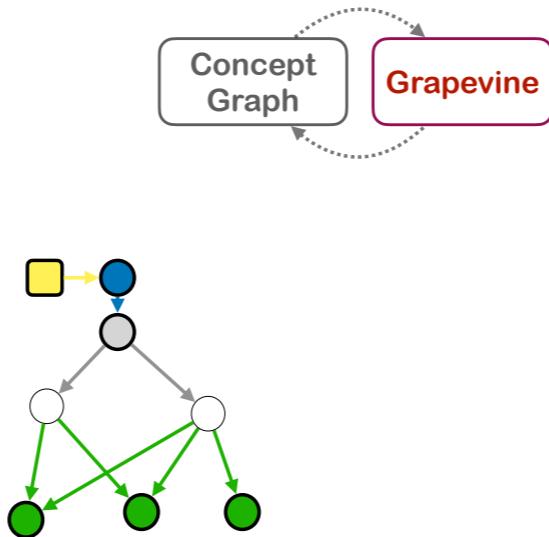
nodes = JSON data files



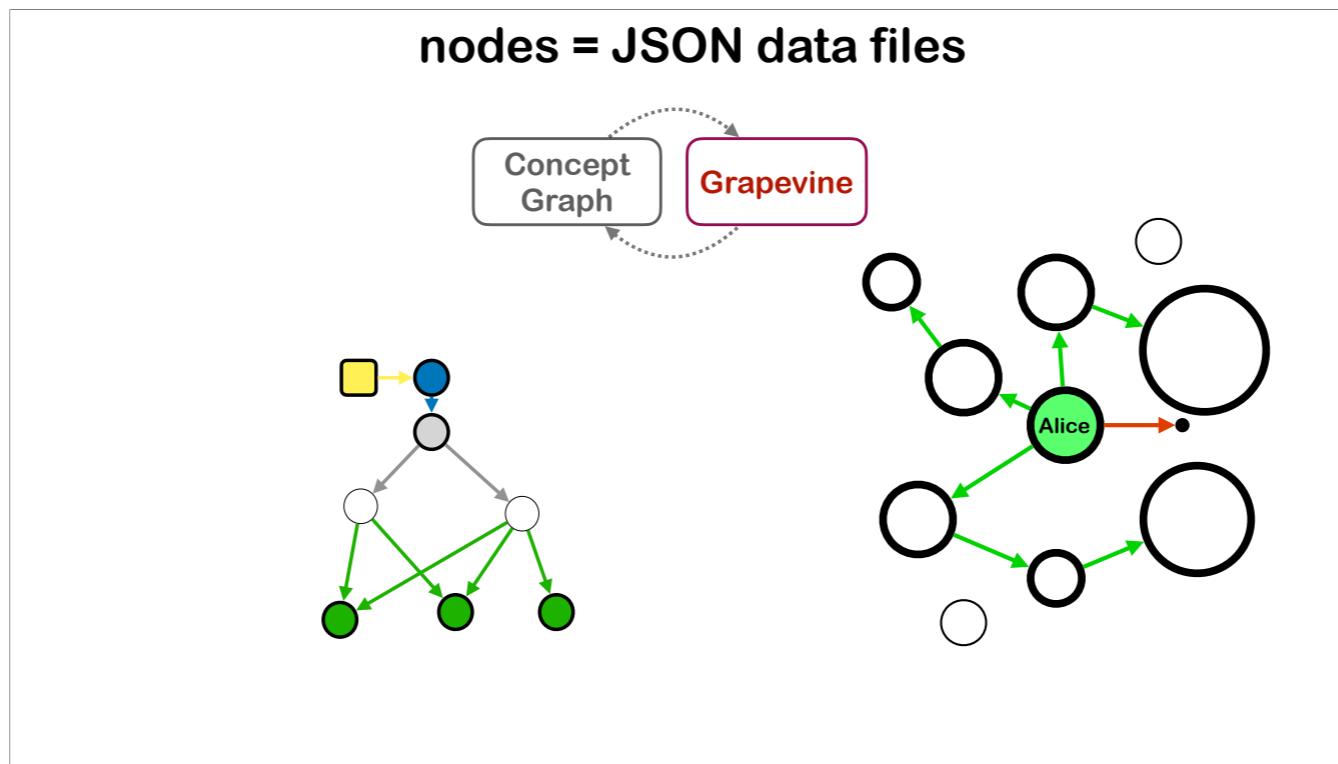
[start 04]

A quick note about some of the visuals I will be showing in this talk. I'm going to be making extensive use of graph databases in both of my apps.

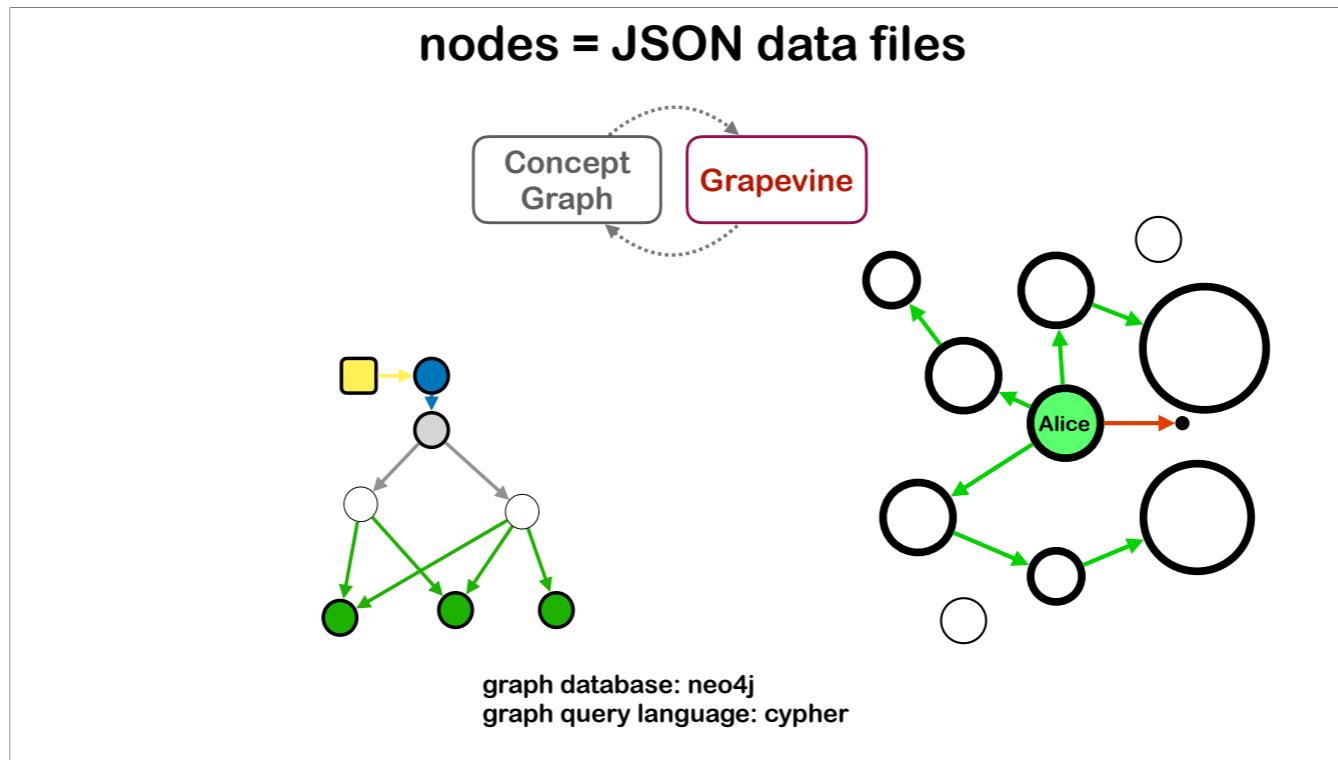
nodes = JSON data files



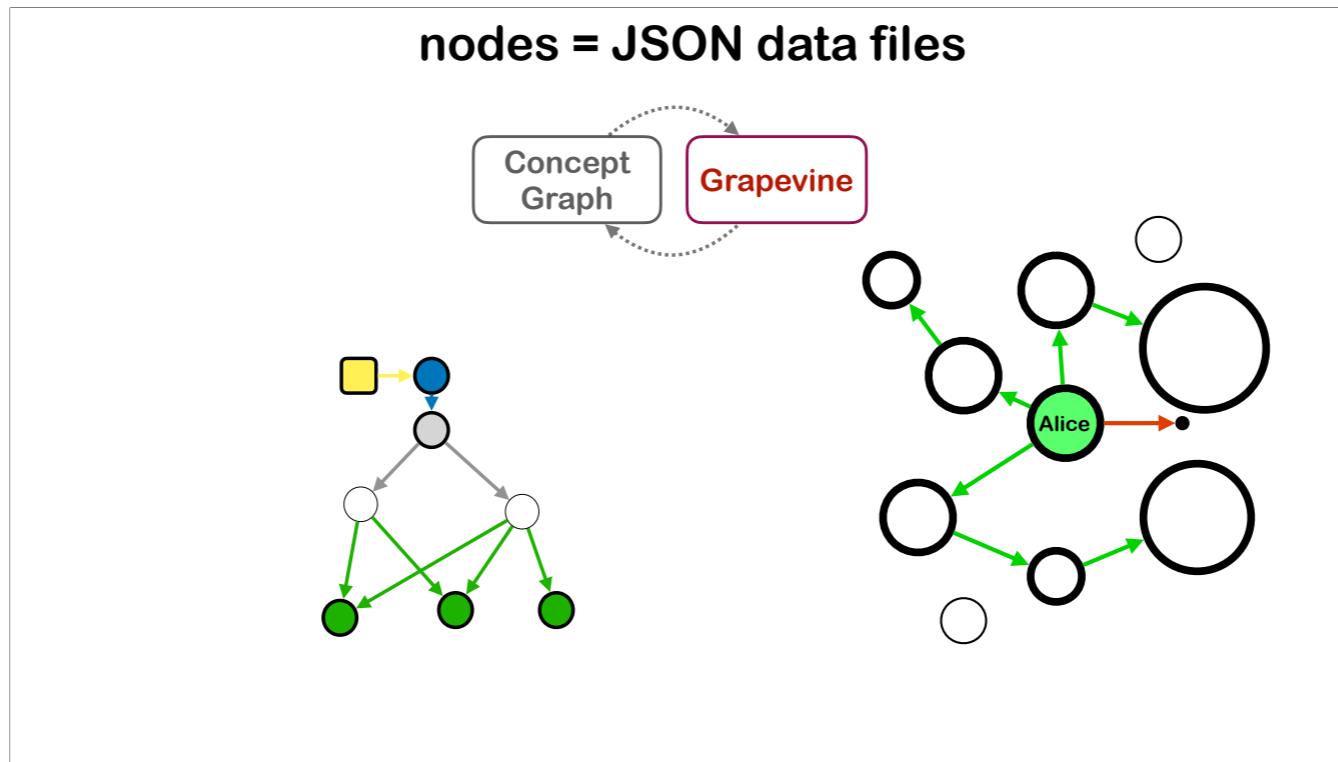
The concept graph will have graphs that look like this, on the left,



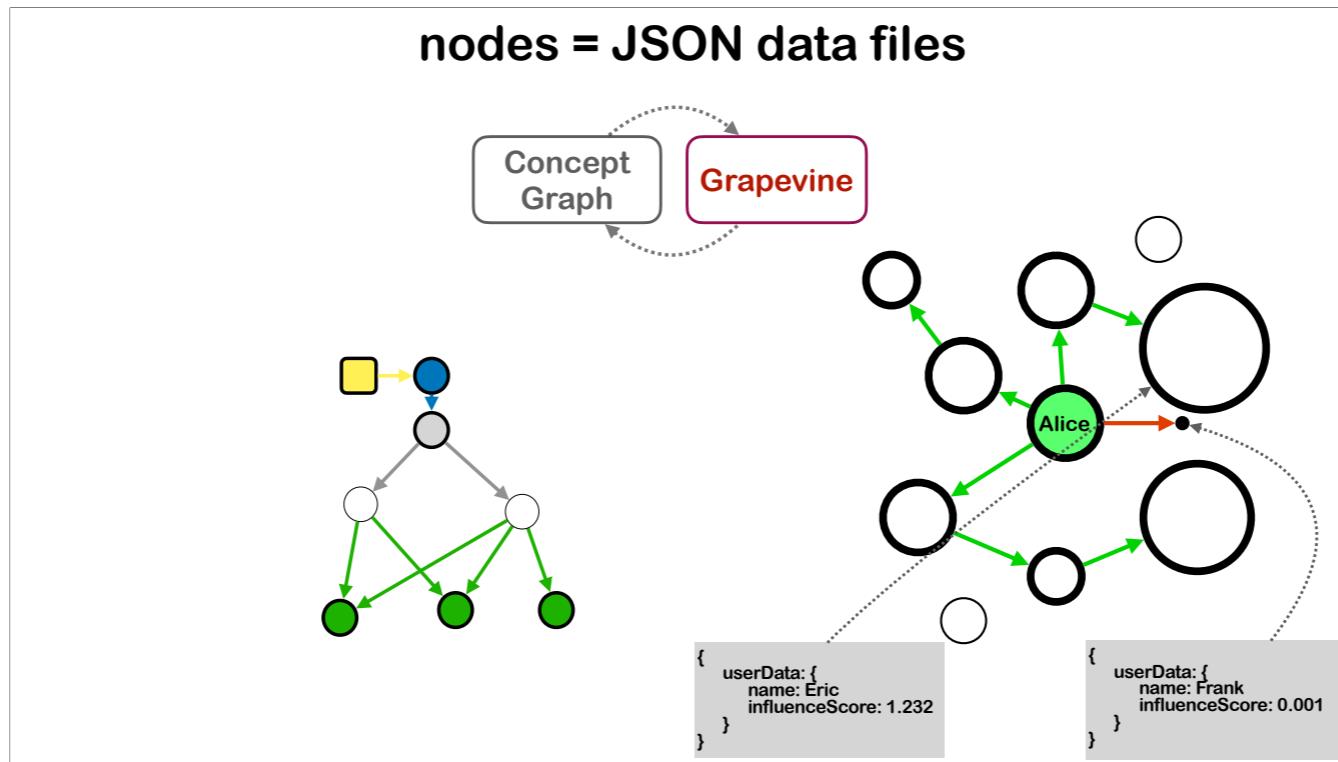
and the Grapevine is going to have graphs that look like this, on the right



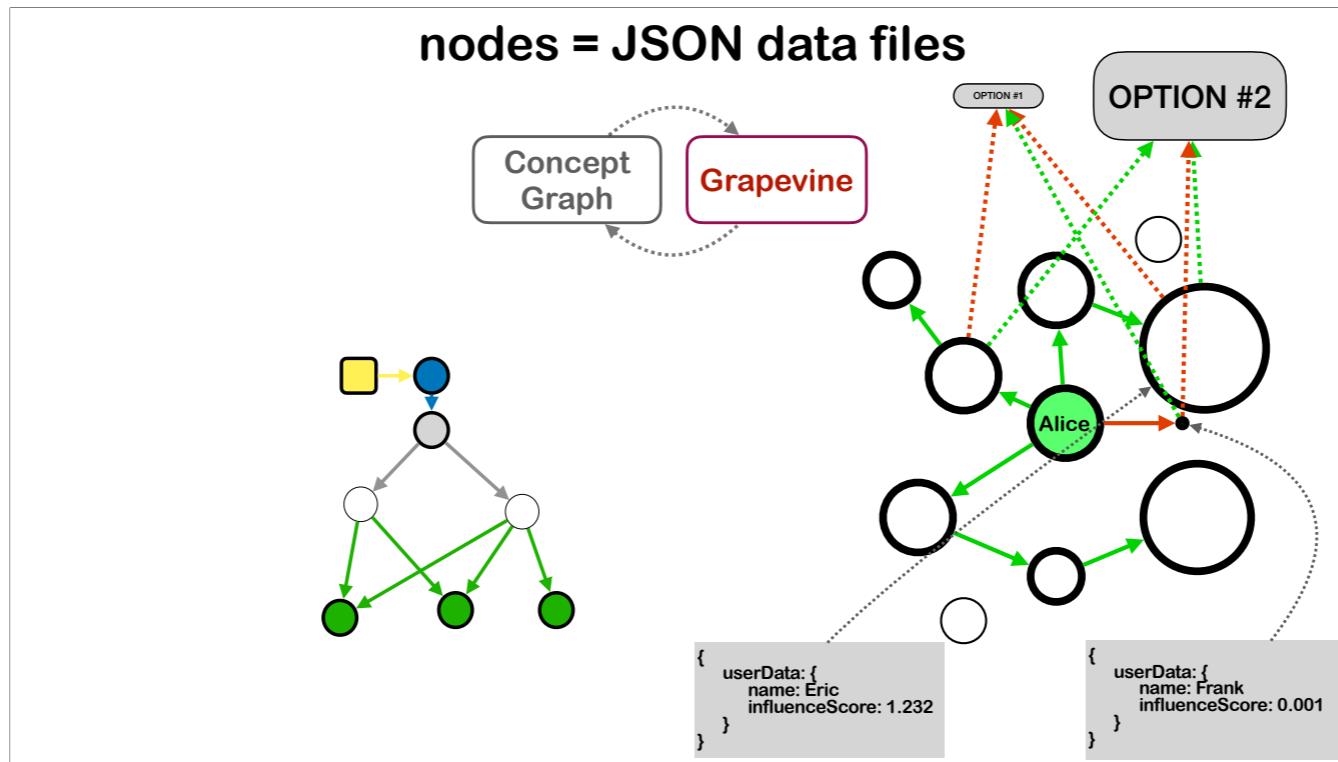
Graph databases (and I'll mention the graph db neo4j and graph query language cypher as tools that I'm familiar with) have nodes that can represent whatever you want and edges, with or without an assigned direction, that represent different types of relationships between the nodes. In my case, whenever I show you a graph, from either app, know that each node (circles or squares or other shapes) corresponds to a JSON data file. That's true in the Concept Graph app on the left and it's true in the Grapevine app on the right.



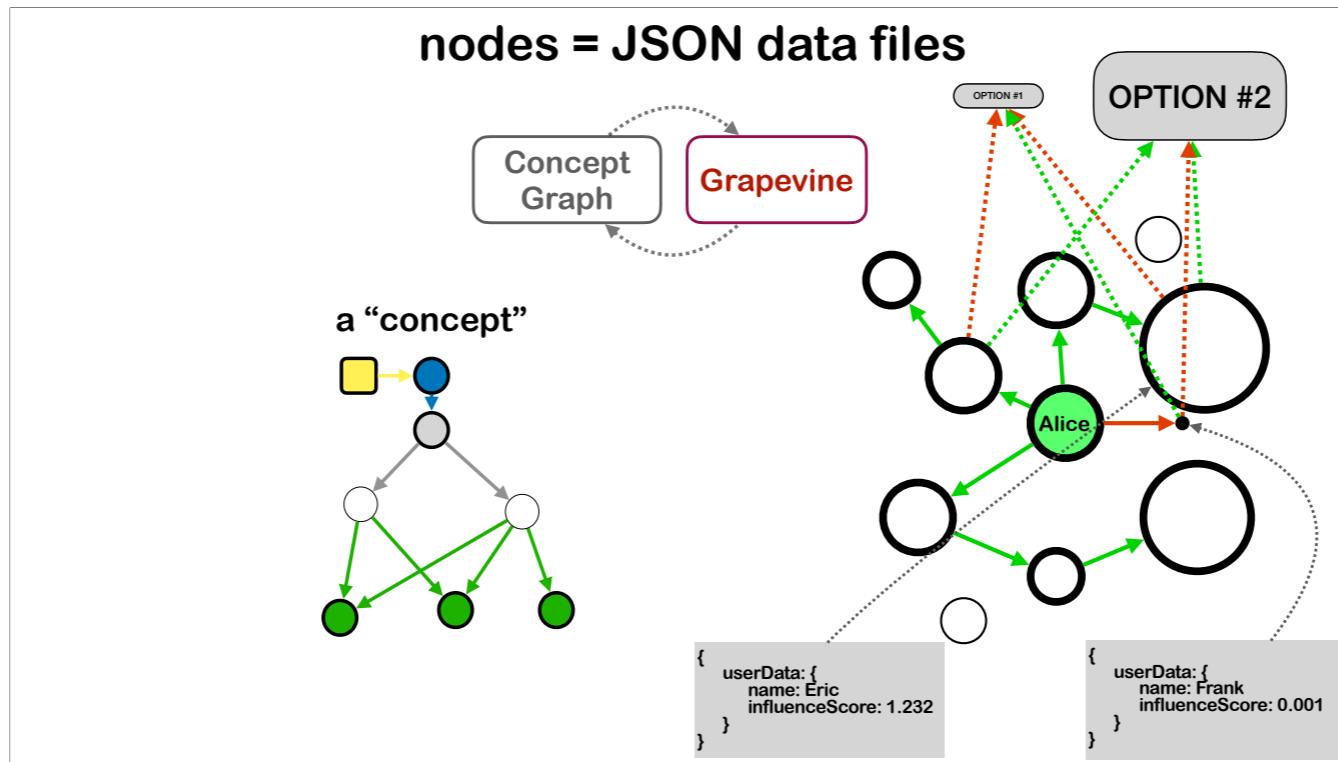
On the right you see a graph that represents Alice's Grapevine. It's a variation on the idea of a web of trust, but it's different from previous attempts that you may be familiar with, in that it is designed and tailored specifically to pair with the CG. Each circle on the right represents a user;



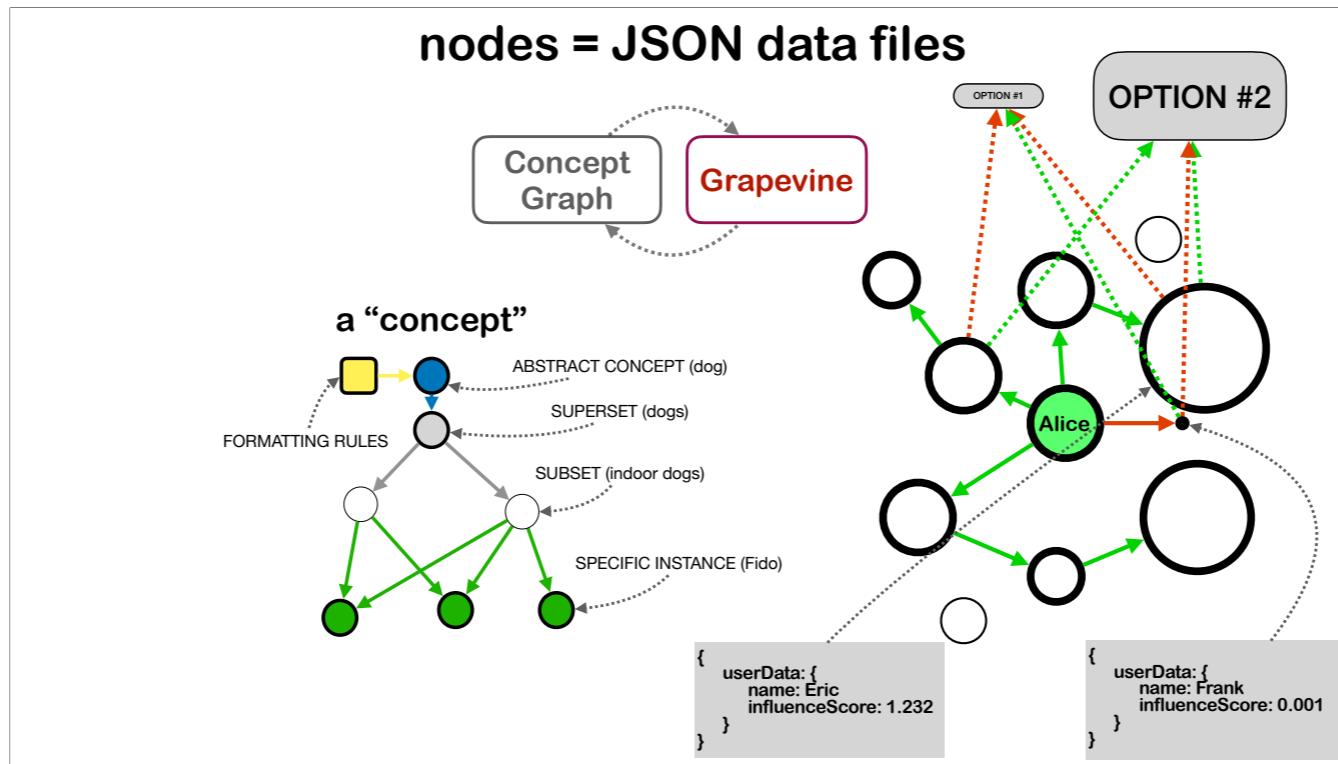
more specifically, each circle represents a JSON data file with information about that user. I'm showing two examples of such JSON files here, in grey. The size of the circles represents how much influence that user wields within Alice's grapevine. Each arrow represents a rating (ideally with a digital signature; we might also call this an attestation) of one user by another user; green favorable, and red unfavorable. There are numbers attached to the ratings but for the sake of simplicity, I'm simply not showing that level of detail here.



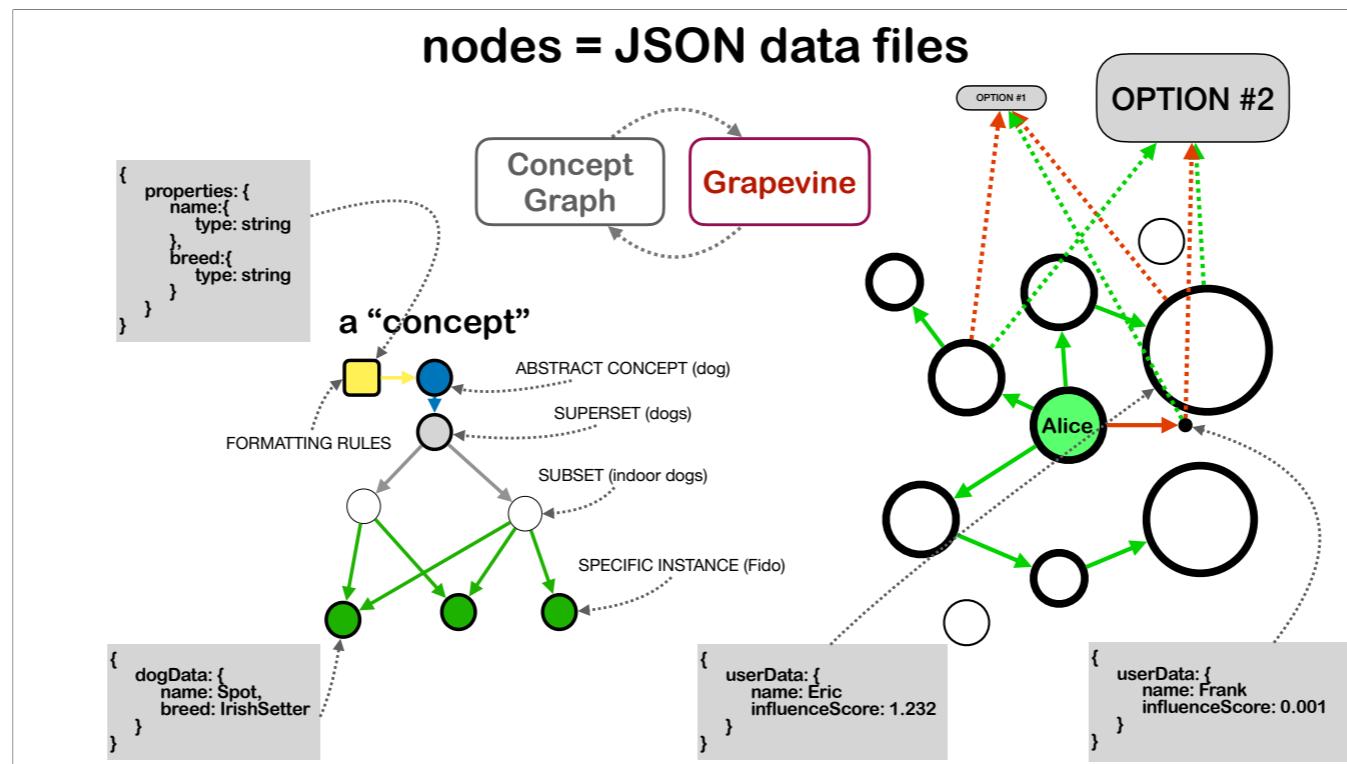
In some cases I will show graphs with nodes that represent things that are being evaluated by the grapevine, anything from: what's the best burger joint, to: what's the best JSON data format to use to represent burger joints. The dotted line arrows, at the upper right of the slide, represent ratings by one user, of one of these various option. As before, each of the OPTION nodes represents a JSON file with information about that option.



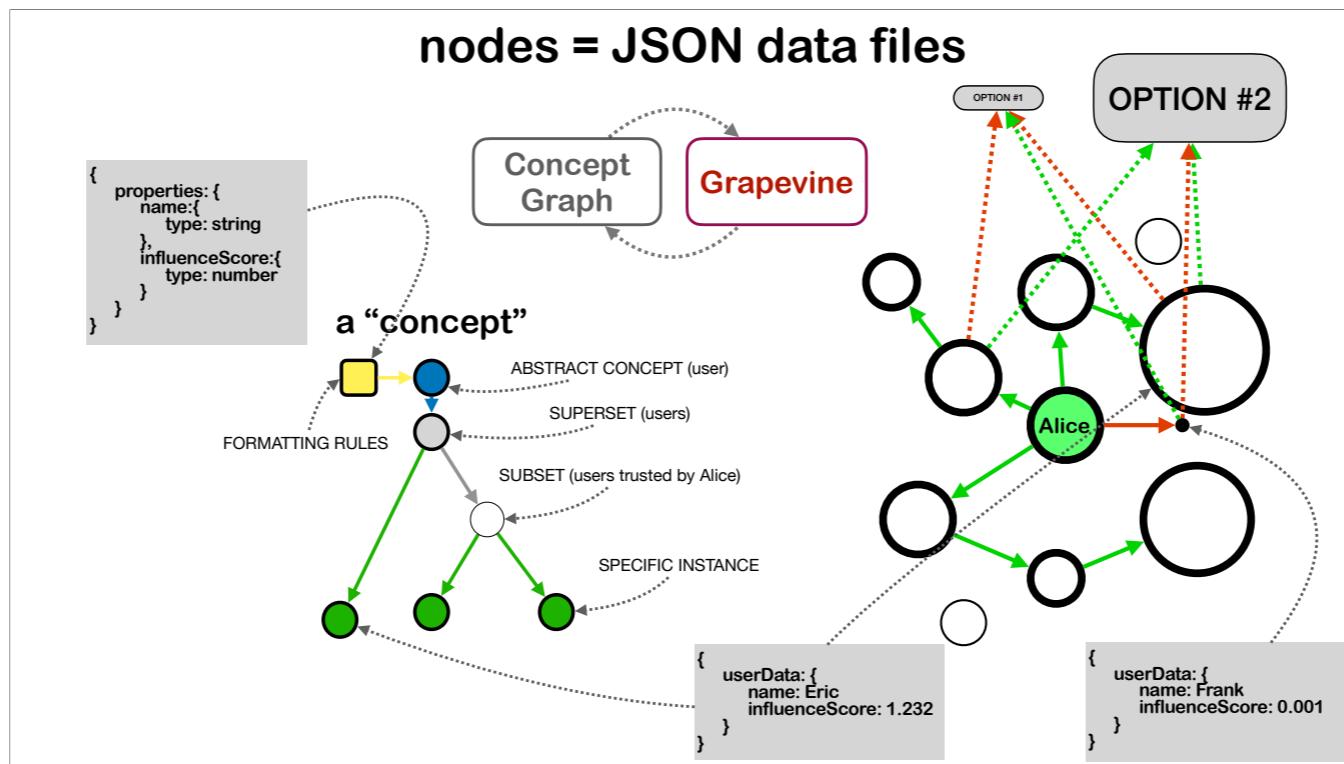
On the left, you see depicted an example of a single concept which is excerpted (from storage locally as part of one of Alice's apps) from the larger concept graph (which typically will contain multiple interrelated concepts).



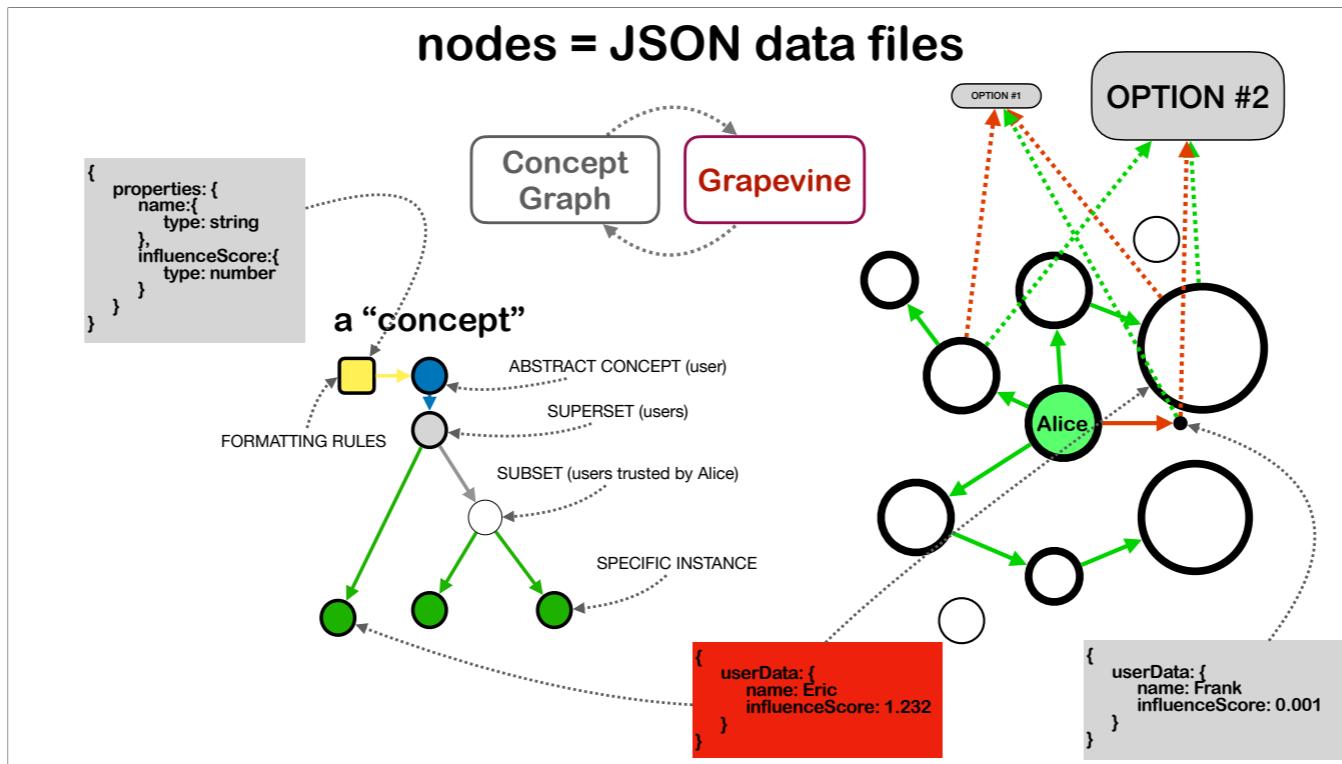
Here, for example, is the concept for dog. The blue node represents the abstract concept of dog; the grey node represents the superset of all dogs in the database; and the green nodes at the bottom represent three specific dogs (with names like Fido, Rover, and Spot).



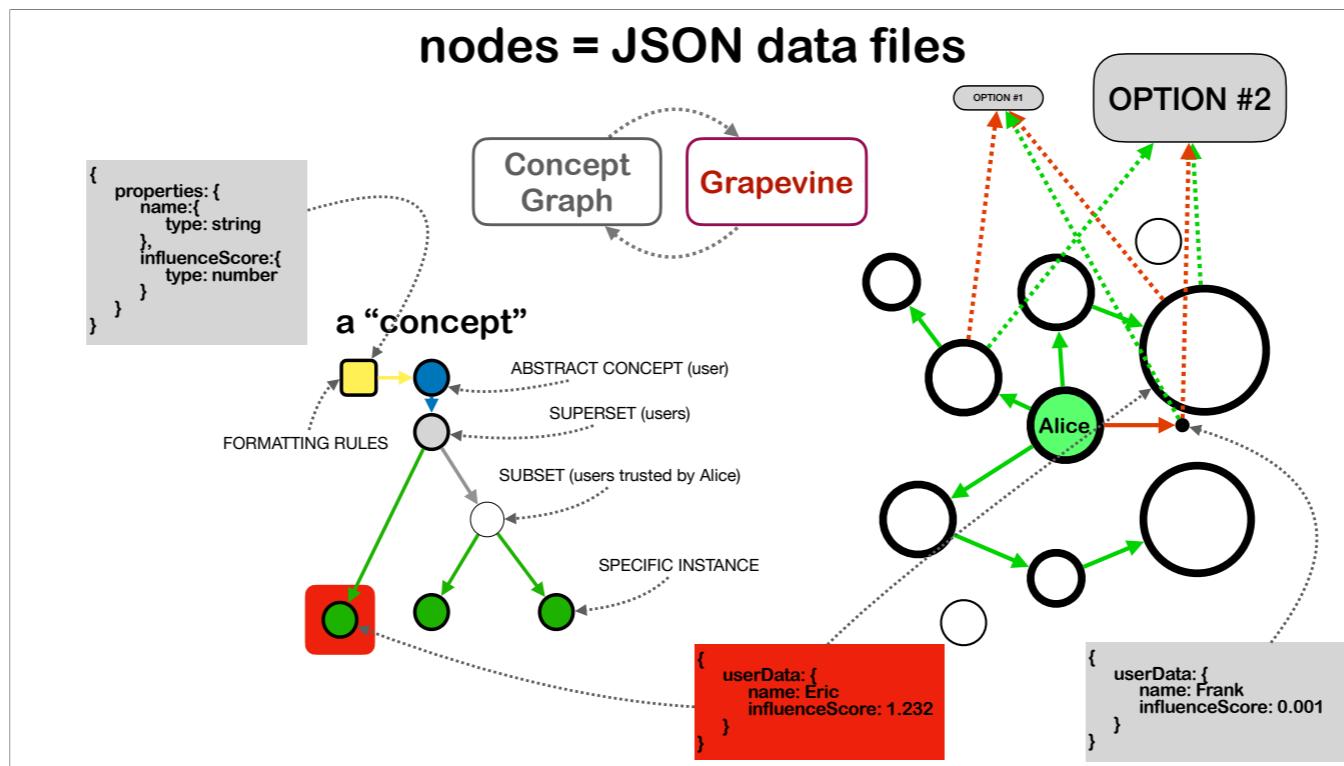
As is the case with the Grapevine, each node in the Concept Graph represents one JSON data file. I'm showing two example JSON files for two of the nodes in the concept on the left. I won't walk though all of the different types of nodes in a CG just yet but I will be going through them later in this video.



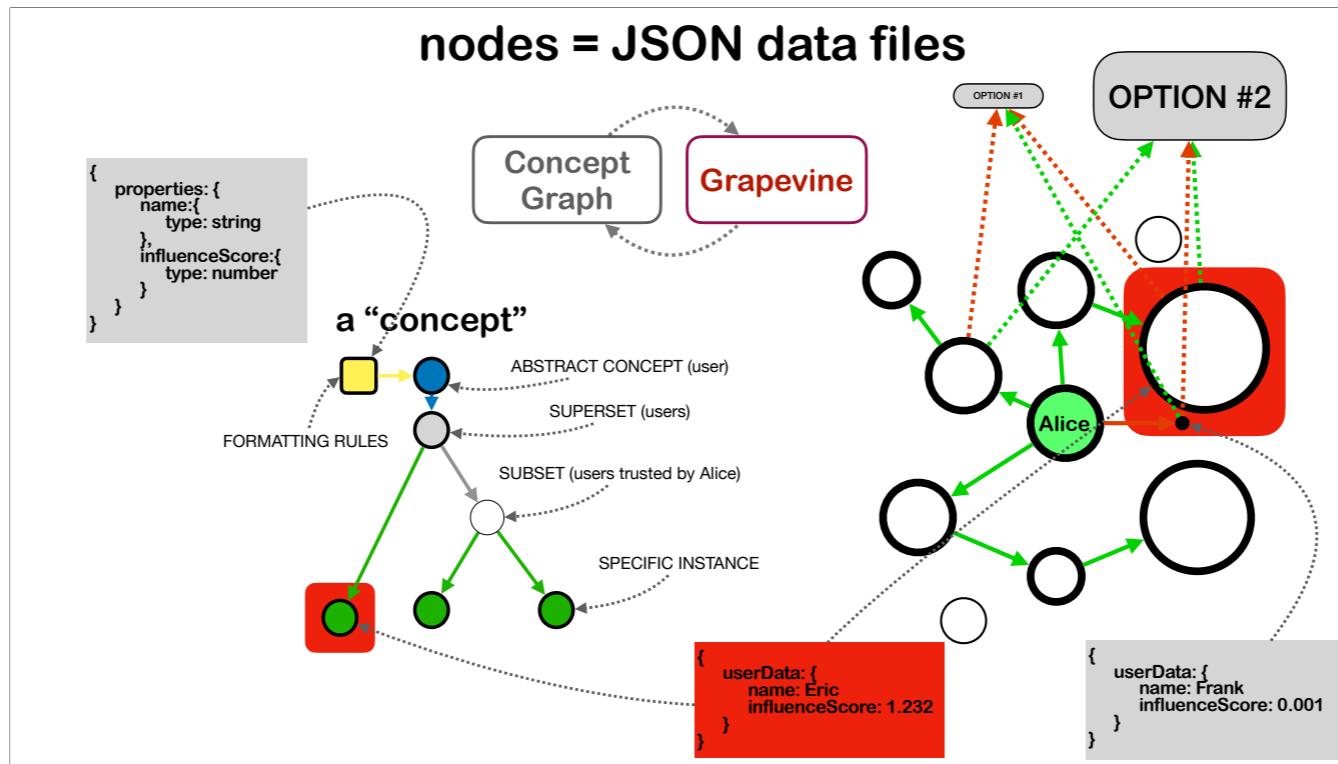
That was the concept for dog; this would be the concept for user. As you can see,



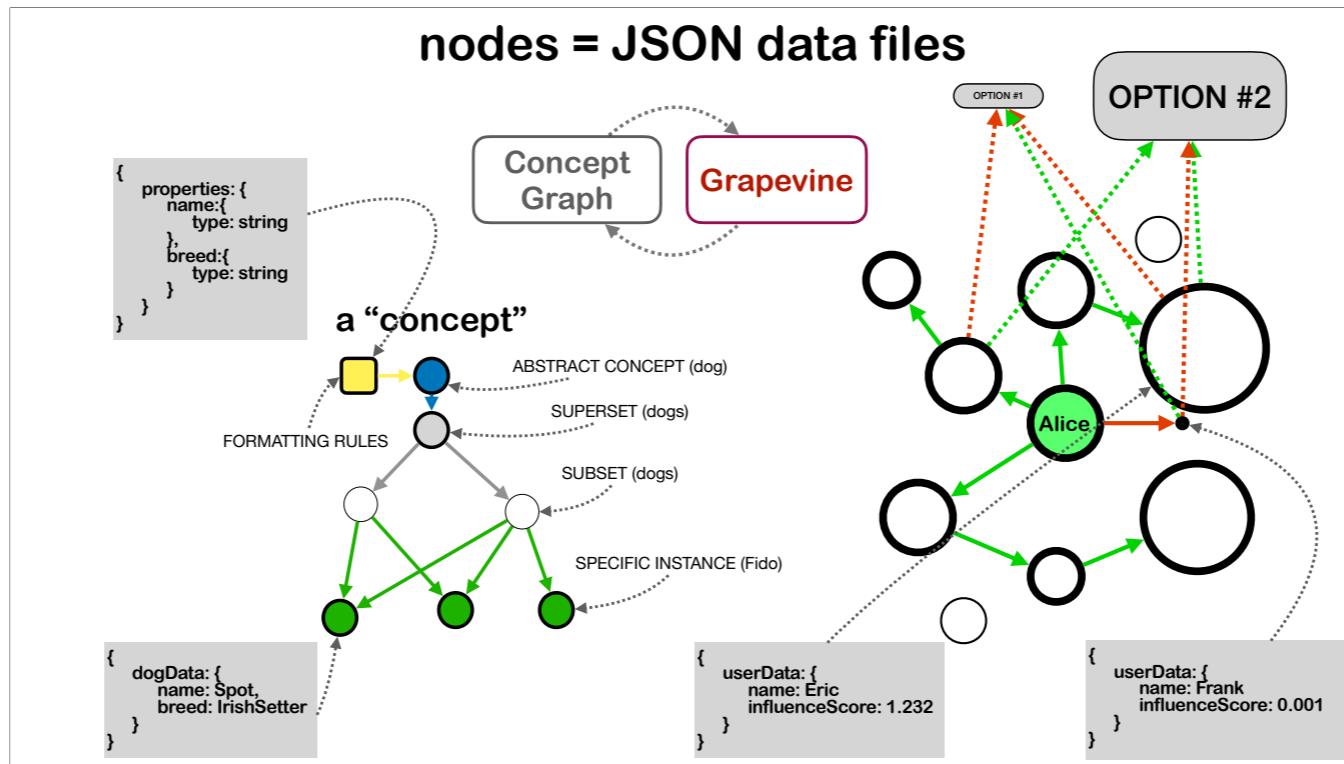
the same file may make an appearance



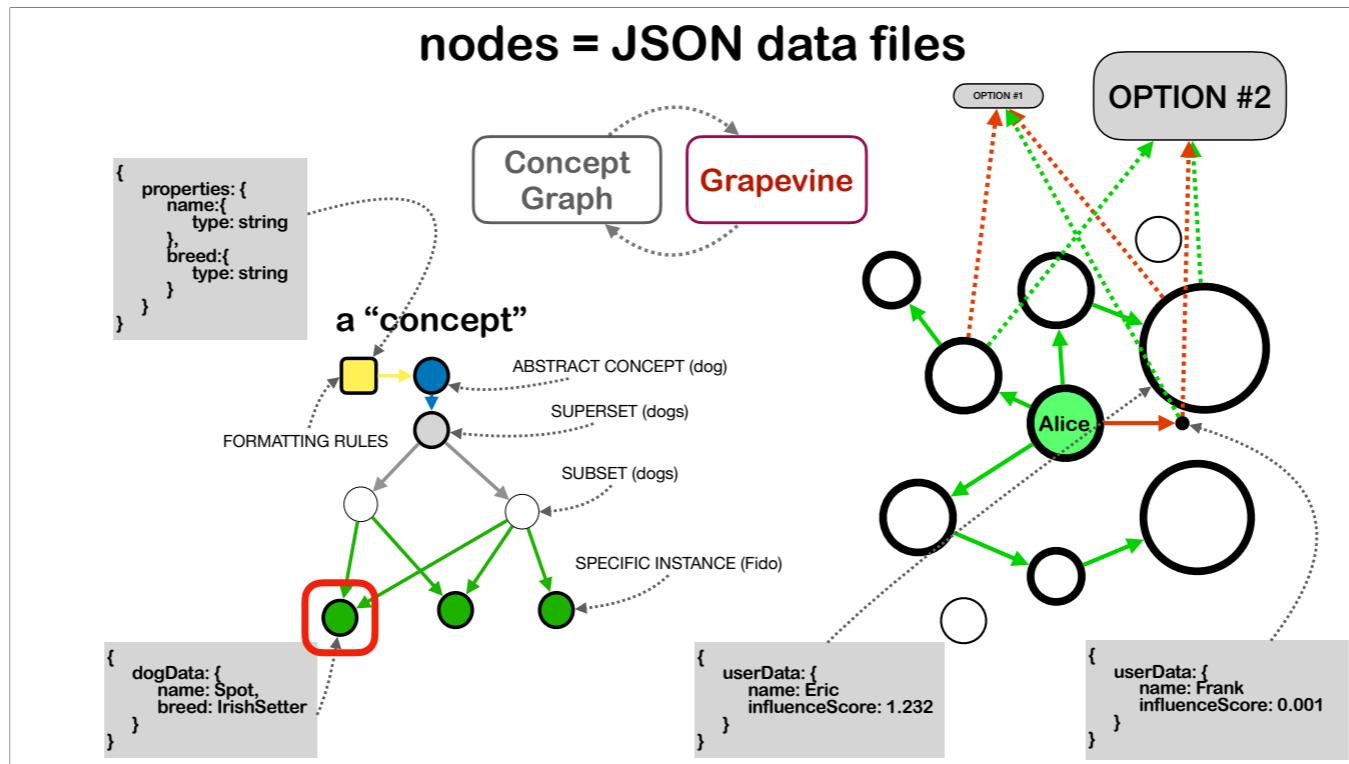
in the CG app on the left



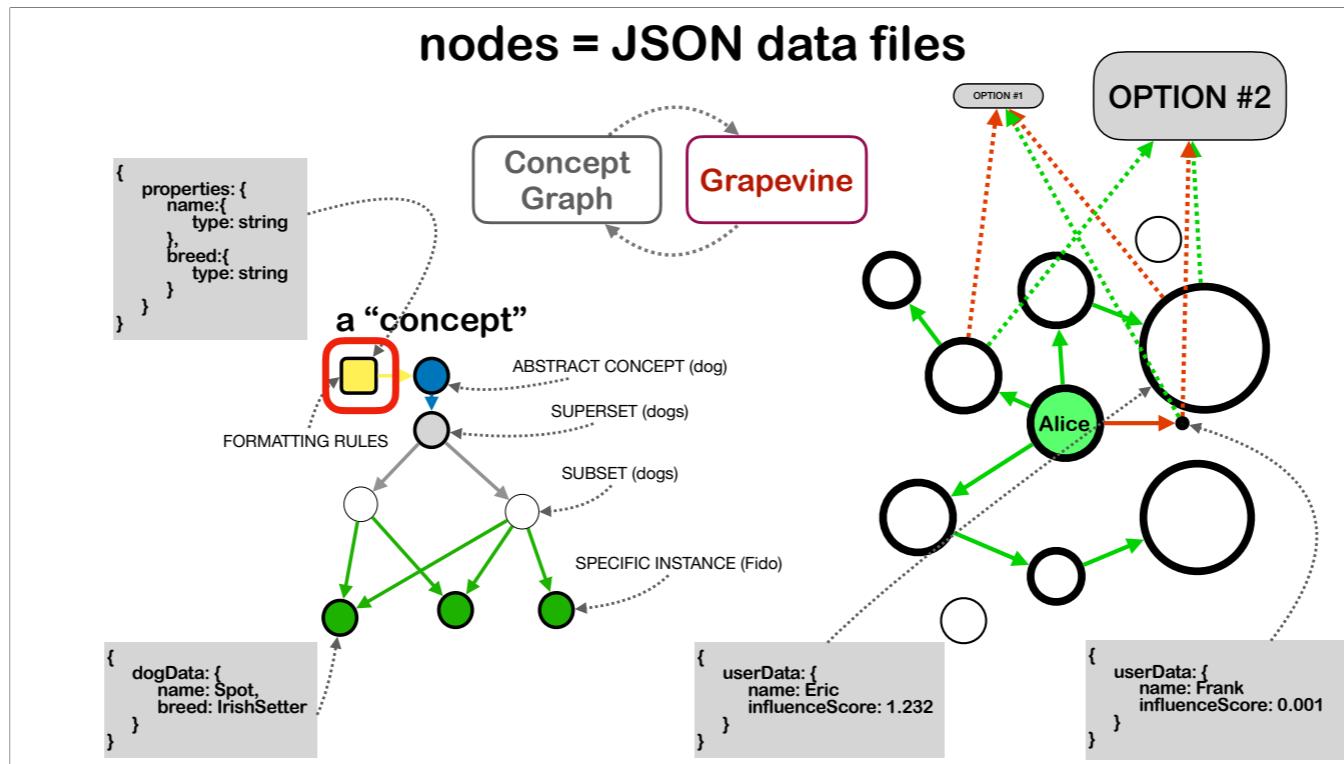
and in the grapevine app on the right, with the same JSON file having one visual appearance in the graph on the left and a different visual appearance in the graph on the right. And that is because different pieces of information are being communicated visually to the user about that file, depending on the purpose of that particular visual representation.



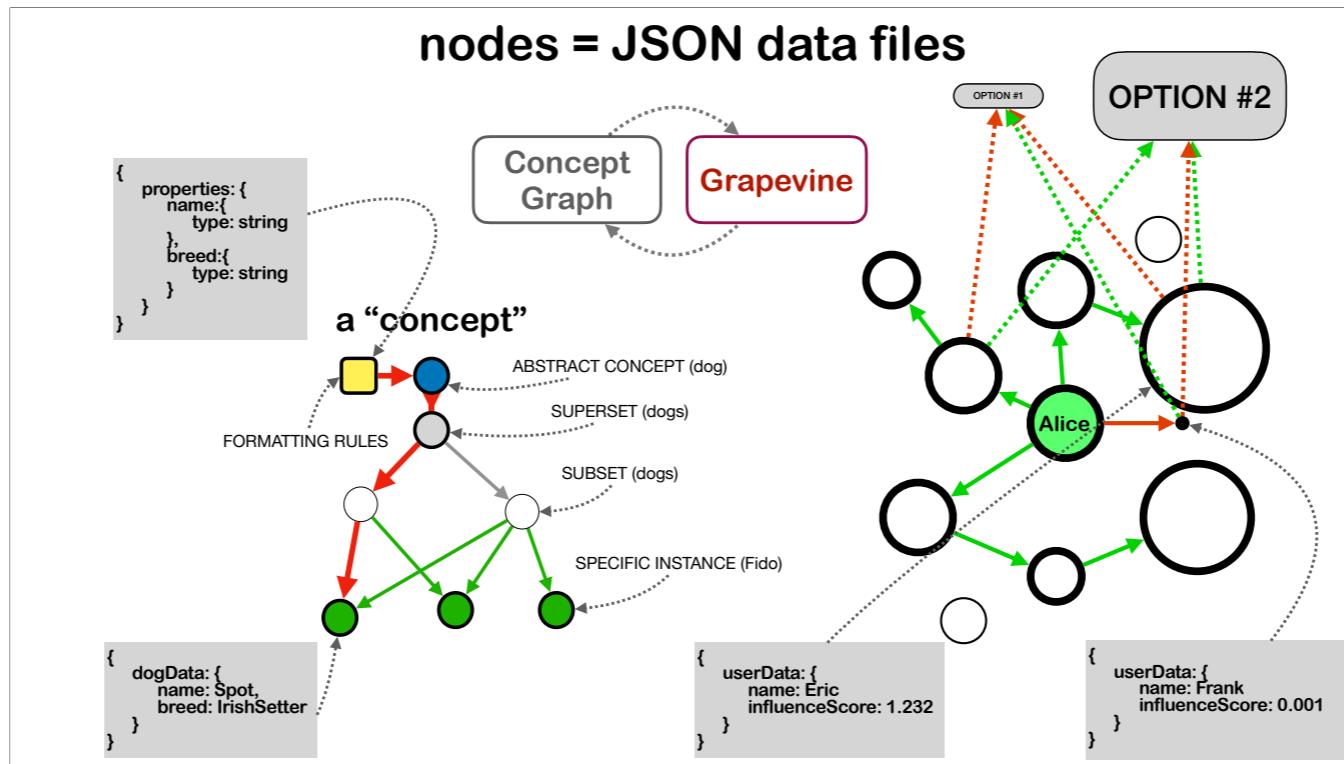
Back to dogs. Each arrow in the CG represents a particular type of relationship between the files (which is why the arrows are different colors), and knowing what those different types of relationships mean, you will be able to trace out the paths which connect any given file,



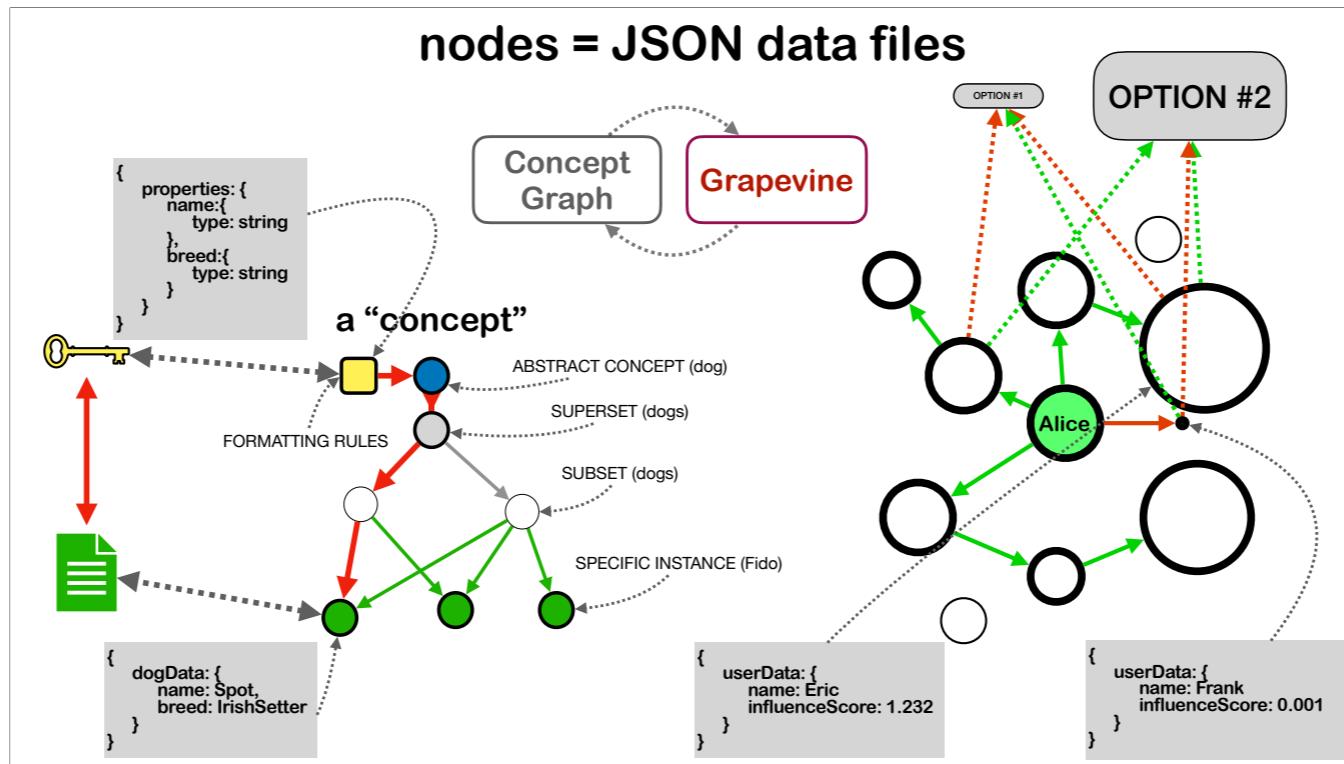
like this one at the bottom of the concept in green,



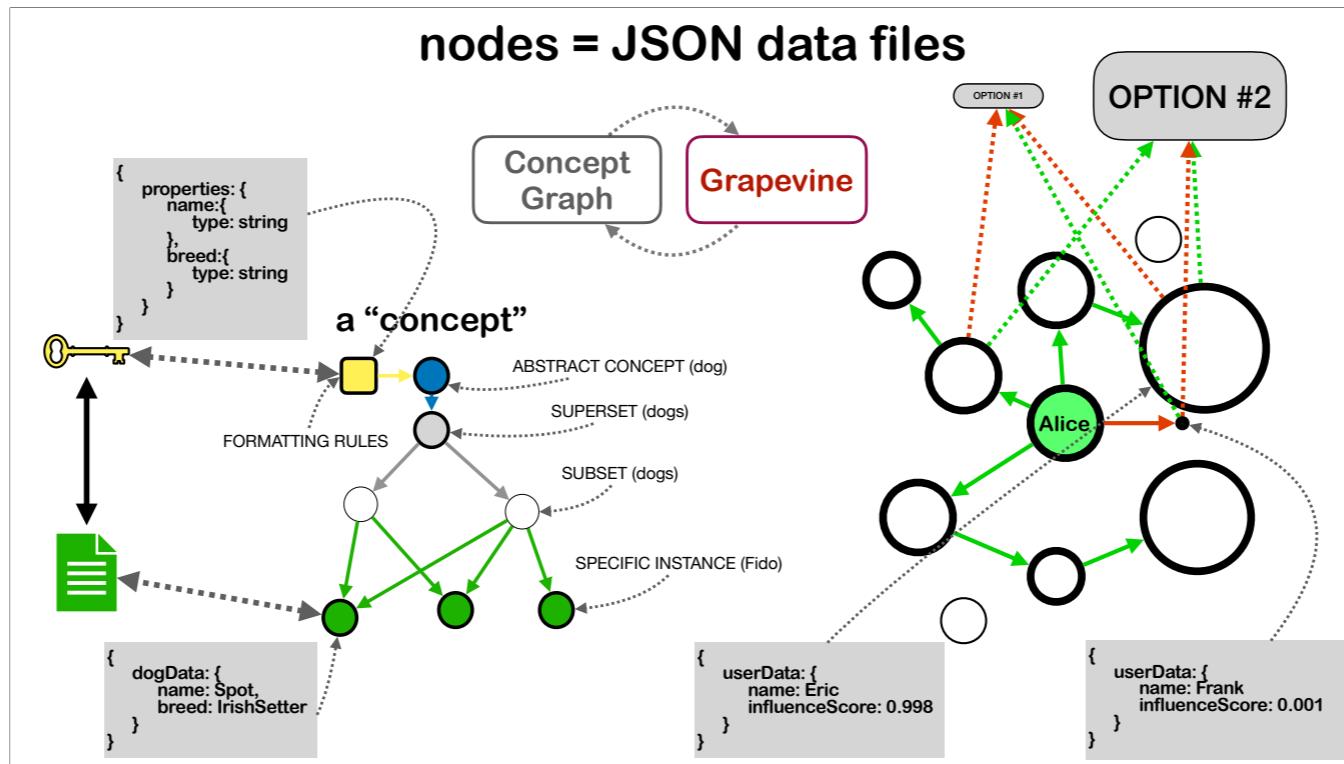
with a specification of its format, like the one at the top of the graph, in yellow,



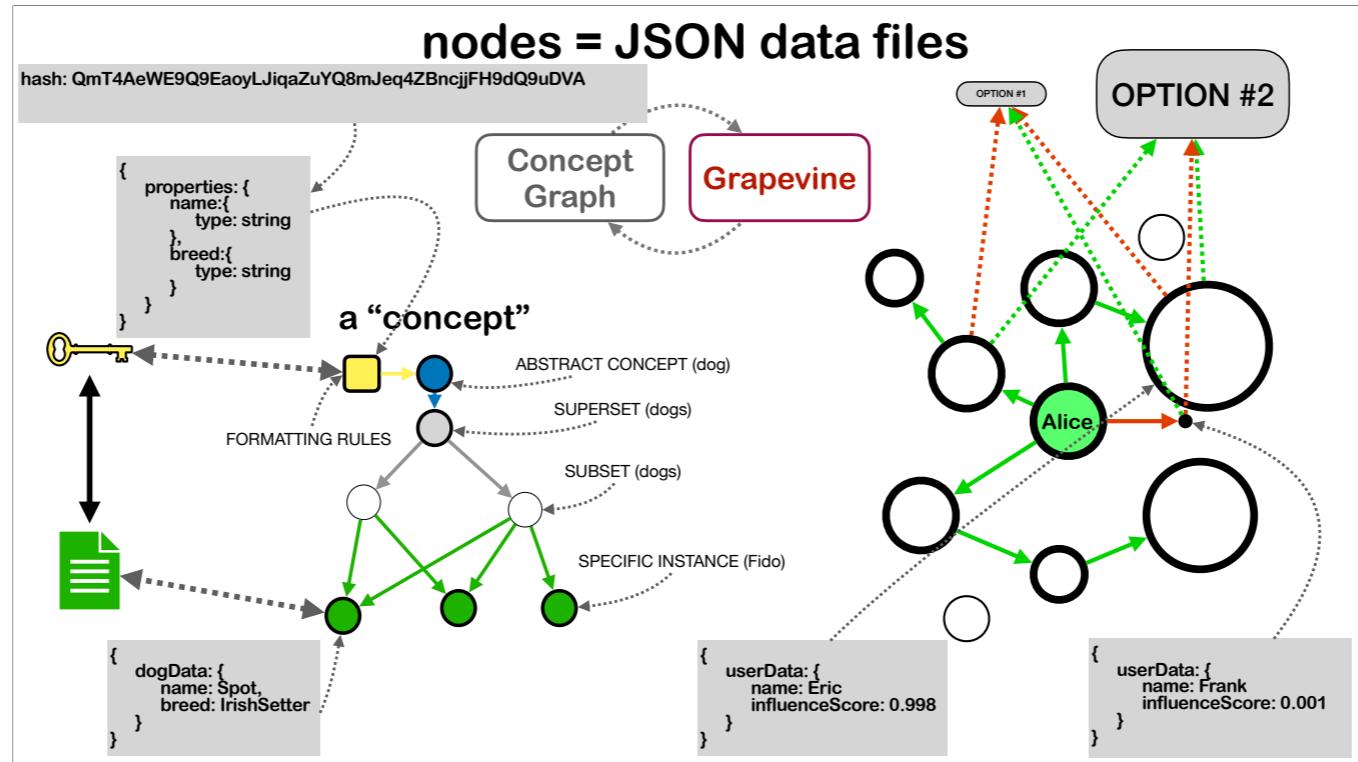
That pathway between them, highlighted here in red, is what I call a Loki Pathway. In this way, by establishing Loki Pathways, we have implemented the Loki Principle,



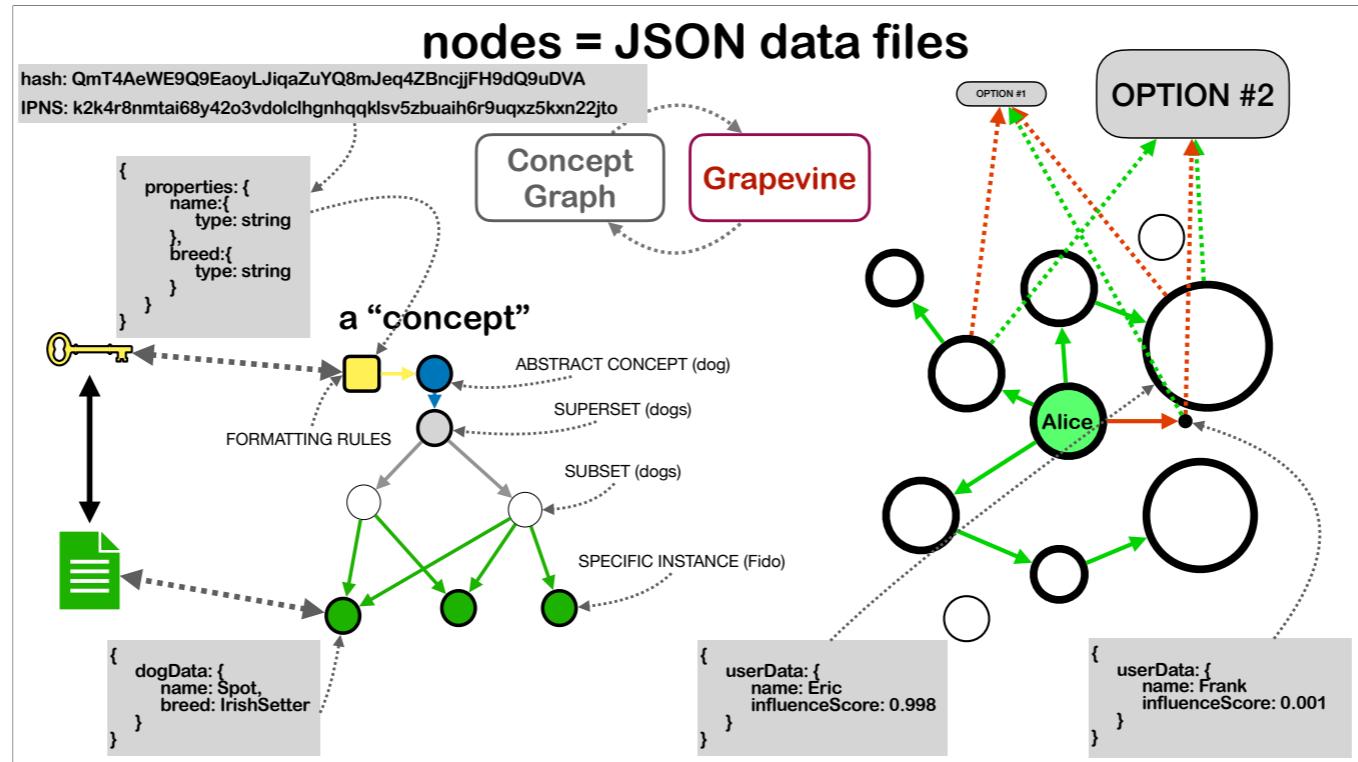
according to which, there must always be a way to connect together any file with a record of its format.



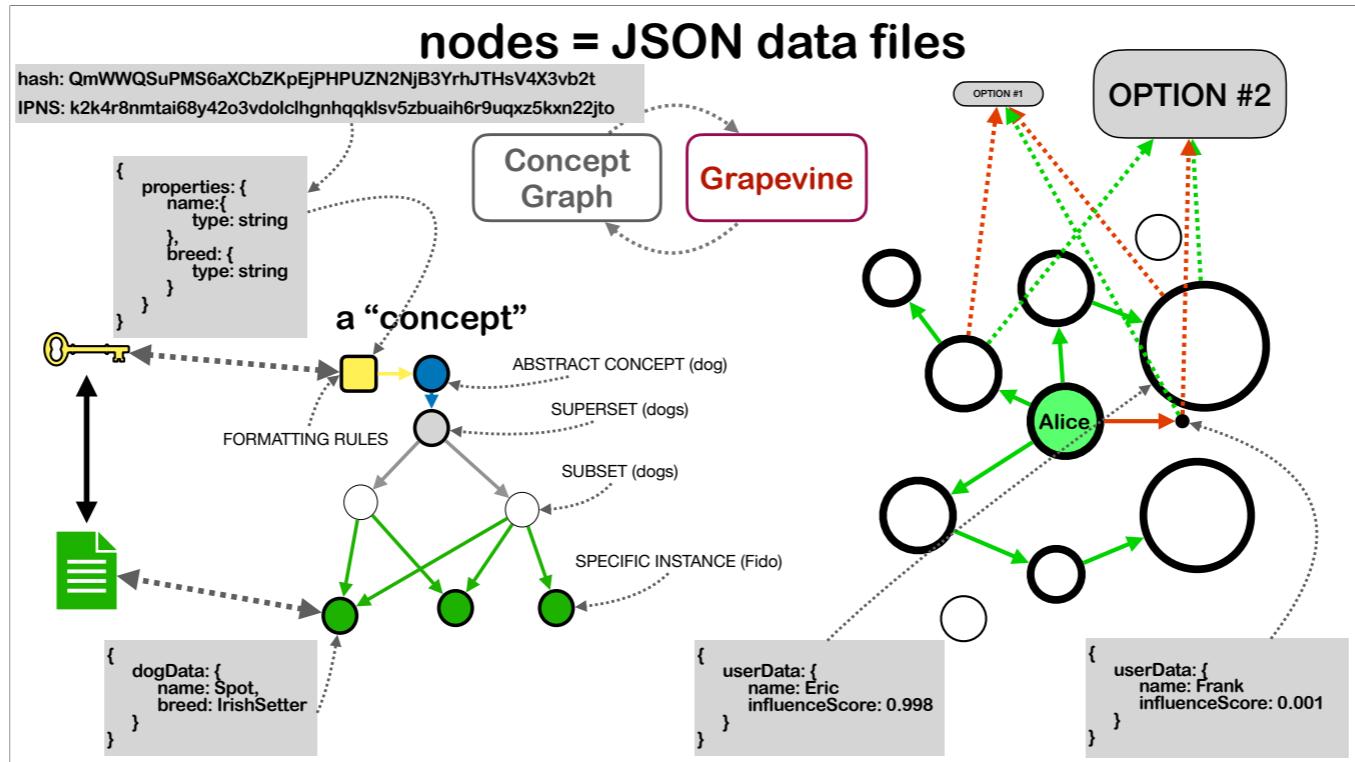
I will review each of these graphs in turn and explain them in further detail later in the video. For now just know that for the CG as well as the Grapevine, each node corresponds to a data file; specifically, a JSON data file.



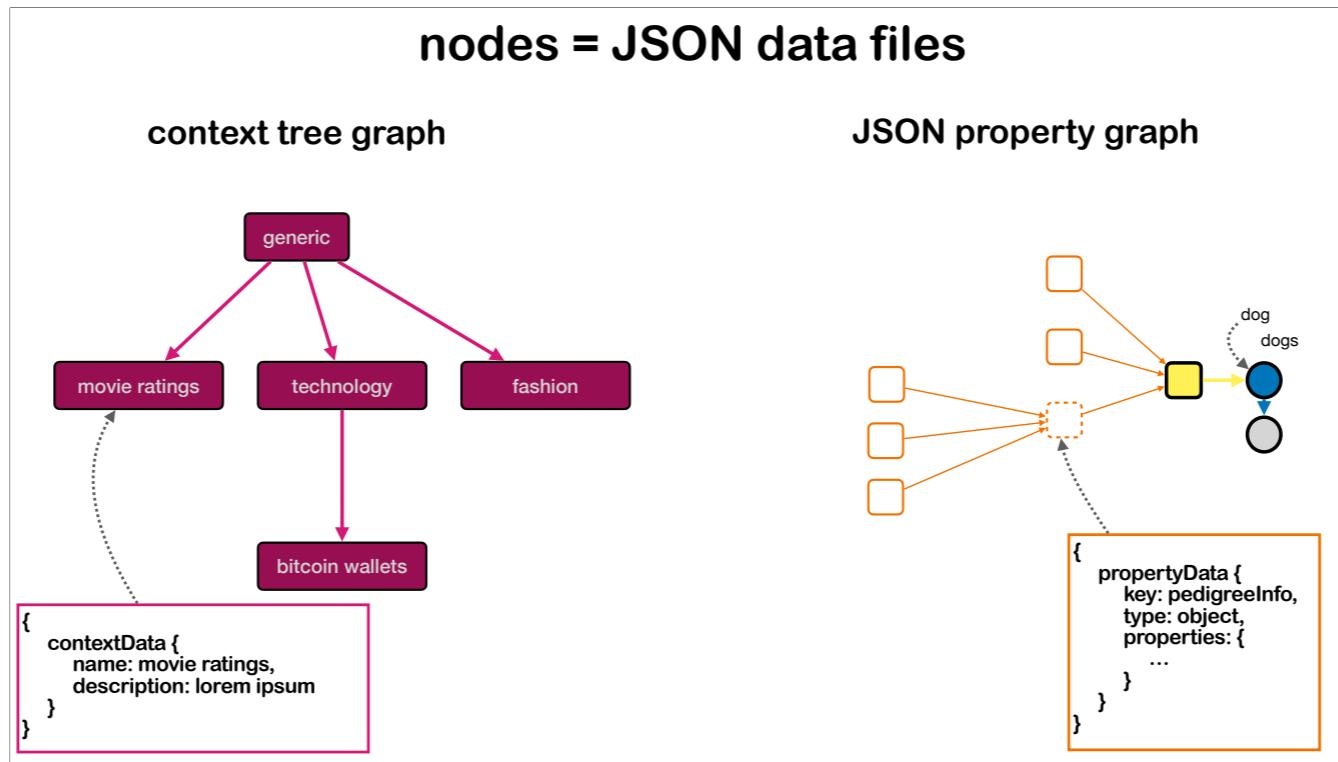
I have adopted the practice, and I got this from working with the IPFS, of using the hash of any given file to refer to that file;



and since the hash changes whenever you make even the tiniest edit, I've adopted the IPNS naming convention, so that any given file has a static IPNS address that is always associated with that file,



even when the file and therefore the hash of the file changes.



There are other types of graphs too. Here are two: a context tree graph on the left, and a JSON property graph on the right (which, as you see, actually connects to one of the nodes in the graph for a concept). I'll explain each of these types of graphs in detail later in this presentation. For now, I'll simply point out that each of these nodes represent a JSON file, with two example JSON files being shown on this slide.

[end 04]

5

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

[start 05]

So, the topics that I'll be covering for the rest of this talk:

I'll review the decentralized Web, what I mean by that phrase, what I mean when I talk about Walled Gardens.

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

I'll review the Principle of Loki.

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

I'll spend quite some time covering the Concept Graph and the Grapevine in some detail.

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

Loose consensus

I'll present how the CG and G are tools for establishing what I am calling Loose Consensus

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

Loose consensus

Cerebral cortex

I'll outline how it is that I think the cortex may ALREADY implement the PoL, and how that could contribute to a model of how knowledge is represented in the brain, something that is still very much an open question.

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

Loose consensus

Cerebral cortex

Progress & Roadmap

I'll review what progress have I made so far, and Roadmap for the future,

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

Loose consensus

Cerebral cortex

Progress & Roadmap

- Phase 1: Loose Consensus on W3C Standards (e.g. Verifiable Credentials)

which includes two phases: phase one being to demonstrate the viability of Loose Consensus by applying these tools to generate consensus on World Wide Web Consortium standards (with specifications for Verifiable Credentials being I think a good place to start);

Topics

The Decentralized Web:

- What is it?
- What do I mean by Walled Gardens

The Principle of Loki

Implementation:

- the Concept Graph
- the Grapevine

Loose consensus

Cerebral cortex

Progress & Roadmap

- Phase 1: Loose Consensus on W3C Standards (e.g. Verifiable Credentials)
- Phase 2: Hybrid Platforms as bridge to the decentralized web

and phase 2, which will be the construction of what I call Hybrid Platforms which will serve as a bridge to facilitate the average internet user, who is not a cypherpunk and not motivated by abstract cypherpunk ideals, to transition from legacy platforms to the platforms of the decentralized web of the future.

[end 05]

6

What is the Decentralized Web?

[start 06]

So, what is the decentralized web?

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

One way to think of it is a replacement for all of the centralized platforms that make up the web as we know it today.

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

Platforms

- Twitter
- Facebook
- eBay
- LinkedIn
- Reddit
- Amazon
- Google
- various dating apps

Examples are shown here on the left panel: Twitter, eBay, Google, etc. These have in common that they are all based on proprietary software, with proprietary algorithms to select what data to feed to us, all centralized around big tech companies.

These platforms allow us to do all sorts of cool things.

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

Platforms

- Twitter
- Facebook
- eBay
- LinkedIn
- Reddit
- Amazon
- Google
- various dating apps

Problems

But there are problems with these platforms.

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

Platforms

- Twitter
- Facebook
- eBay
- LinkedIn
- Reddit
- Amazon
- Google
- various dating apps

Problems

- surveillance capitalism
- disinformation
- fake user accounts
- ratings farms
- censorship and deplatforming
- poisoned public discourse

Problems like: surveillance capitalism; disinformation, what some would call fake news, spurred by motivations that we sometimes can only guess at; fake user accounts; ratings farms that significantly degrade the believability and utility of ratings at sites like Amazon or Yelp; censorship; deplatforming; the poisoning of public discourse, impacted by algorithms for content selection that we are not privy to, that we don't understand — heck even the companies themselves don't even understand! — but which invariably have the effect of getting us all angry with each other. And for no good reason; except that it's good for advertising.

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

Platforms

- Twitter
- Facebook
- eBay
- LinkedIn
- Reddit
- Amazon
- Google
- various dating apps

Problems

- surveillance capitalism
 - disinformation
 - fake user accounts
 - ratings farms
 - censorship and deplatforming
 - poisoned public discourse
- Why?
- advertising: perverse incentives
 - algorithms that manipulate our emotions

Why does this happen? One could consider the underlying business model, which relies heavily on advertising. This creates perverse incentives regarding the way that these platforms are designed. It leads to algorithms that choose content in ways that are ever more effective at keeping our eyes glued to the screen, maximizing ad revenue, but at the cost of being emotionally, spiritually, even cognitively damaged.

What is the Decentralized Web?

Replacement for centralized platforms & Big Tech

Platforms	Problems
<ul style="list-style-type: none">• Twitter• Facebook• eBay• LinkedIn• Reddit• Amazon• Google• various dating apps	<ul style="list-style-type: none">• surveillance capitalism• disinformation• fake user accounts• ratings farms• censorship and deplatforming• poisoned public discourse <p>Why?</p> <ul style="list-style-type: none">• advertising: perverse incentives• algorithms that manipulate our emotions <ul style="list-style-type: none">• we don't control our data

But ultimately, I think these problems stem from the fact that we do not really control our data. We have yielded control to these large centralized institutions. Voluntarily. For free. Why? bc we do not have better alternatives. At least, not YET. And that's why I'm doing what I'm doing: bc I envision the day when we, as users, are able to exercise exquisite control over our data: social media data, financial data, yes even medical data. We need to be able to fine tune privacy; but at the same time, it needs to be user friendly; something anyone — not just developers, not just idealistic cypherpunks — anyone will want to use, and will be able to use. These goals have seemed out of reach for too long. But I believe we'll see how they can come to be within our grasp once we add the methods that I am outlining in this video to the many other tools that we already have.

Examples of Tools for the Decentralized Web

IDENTITY:

- PGP (Phil Zimmermann, 1991)
- Self Sovereign Identity (SSI) / Decentralized Identifier (DID)

WEB OF TRUST

- 1992, Zimmermann
- Rebooting the Web of Trust (RWOT, Christopher Allen)

DATA STORAGE

- decentralized data storage (e.g. IPFS, GUN)

PAYMENTS

- BTC, lightning network

OPEN SOURCE PLATFORMS

- OpenBazaar (attempt at a decentralized version of Amazon or eBay)

Traction: limited. Why?

What other tools am I talking about? Many of them are designed with privacy in mind and as such rely heavily upon cryptography. They are what I sometimes call the cypherpunk toolkit. Here is a sampling of some of them on this page. I am not going to walk through these individually. This page is not at all intended to be complete.

Traction in terms of usage on all of these (with the possible exception of bitcoin) has been WAY less than what many of us would like it to be, as evidenced by the fact that we are all still using the proprietary, centralized platforms on the previous slide. Why is that? Different people will have different ways of answering this question, but as I've said earlier, the problem I am focused on is the problem of Walled Gardens.

[end 06]

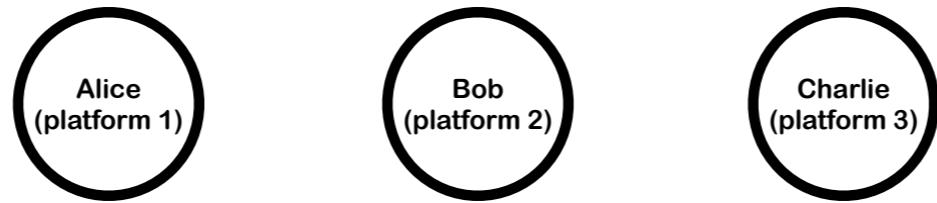
7

Walled Gardens

[start 07]

What do I mean by Walled Gardens?

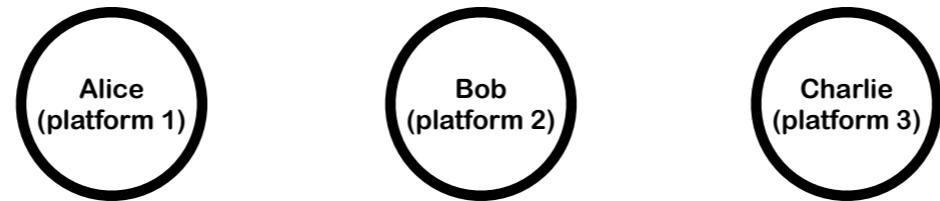
Walled Gardens



Consider that one of the central goals of dWeb, at least as I see it, is portability, which means, users, like Alice, Bob, and Charlie here, ought to be able to communicate online in a privacy preserving manner, without necessarily having to use the same platform. As you can see here, Alice, Bob, and Charlie are each on platforms 1, 2, and 3, respectively. Everything that we can do using centralized platforms, but which typically require any two users to be on the same platform, we ought to be able to do using dWeb, WITHOUT USERS BEING ON THE SAME PLATFORM. Because if you expect and require people to be using the same platform, then you have given up on portability. And that's not acceptable. You cannot expect everyone to agree on which platform to use; on what type of blockchain to use. Or what type of DID or SSI to use. Or what type of decentralized data storage system to use. Or even what type of payments system to use. I think BTC/LN is the way to go for payments; but what if I'm wrong? or what if not everyone on the planet agrees with me? Just like BTC, it is often said, is for enemies; the dWeb needs to accommodate users even as they are working from disconnected and disparate toolsets.

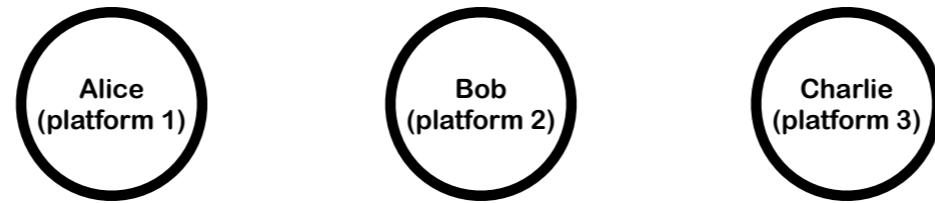
So: let's assume Alice, Bob, and Charlie exist here on 3 different platforms (meaning: 3 different applications, made by competing groups of developers); they use different tools for identity, data storage, etc.

Walled Gardens



If these are proprietary, centralized, legacy platforms, they can't really communicate very effectively with each other. Data from one platform is not portable to another platform. For example: suppose Alice opens a new account on eBay. Suppose she has older, existing accounts in Reddit, facebook, linkedIn. But can she port her reputation from those platforms onto eBay? No, she cannot. Which means that she needs to build her rep as a trusted seller starting at ground zero.

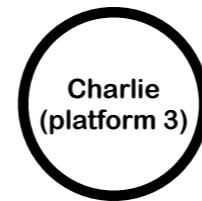
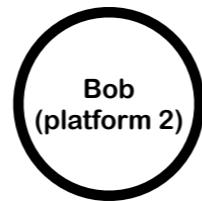
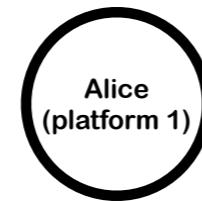
Walled Gardens



Why is that? Well, one thought might be that it is not in the economic interests of individual proprietary platforms to make it easy to export data in a format that would make it readily useful to other competitor platforms.

Walled Gardens

replace proprietary platforms with open source

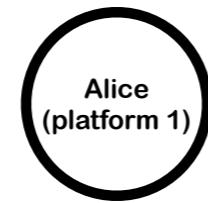


So the first solution that might come to mind would be to replace proprietary platforms with open source platforms.

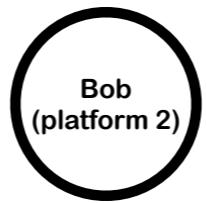
Walled Gardens

replace proprietary platforms with open source

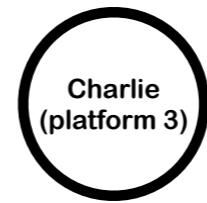
decentralized
eCommerce
(Amazon/eBay)



decentralized
social media
(Twitter)



decentralized
search
(Google)

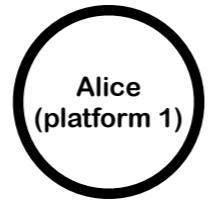


For example: decentralized eCommerce (OB1's Open Bazaar being an example of this); or a decentralized Twitter (again, OB1's Haven is an example of this); or a decentralized Google (a good example of this does not yet exist as far as I am aware).

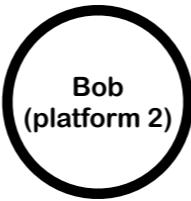
Walled Gardens

replace proprietary platforms with open source

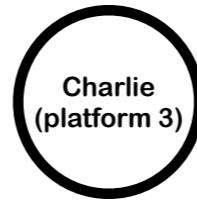
decentralized
eCommerce
(Amazon/eBay)



decentralized
social media
(Twitter)



decentralized
search
(Google)



github account #1

github account #2

github account #3

This means that the code for these platforms is public and readily accessible from some place like GitHub or some other public repository. Data is no longer necessarily hidden in centralized proprietary platform servers where even the users themselves do not have access to all of their own raw data.

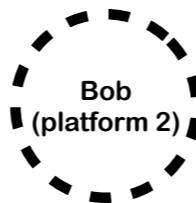
Walled Gardens

replace proprietary platforms with open source

decentralized
eCommerce
(Amazon/eBay)



decentralized
social media
(Twitter)



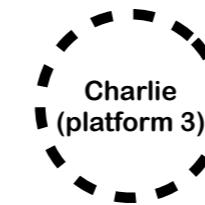
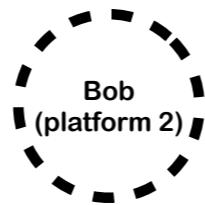
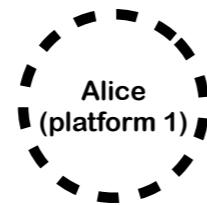
decentralized
search
(Google)



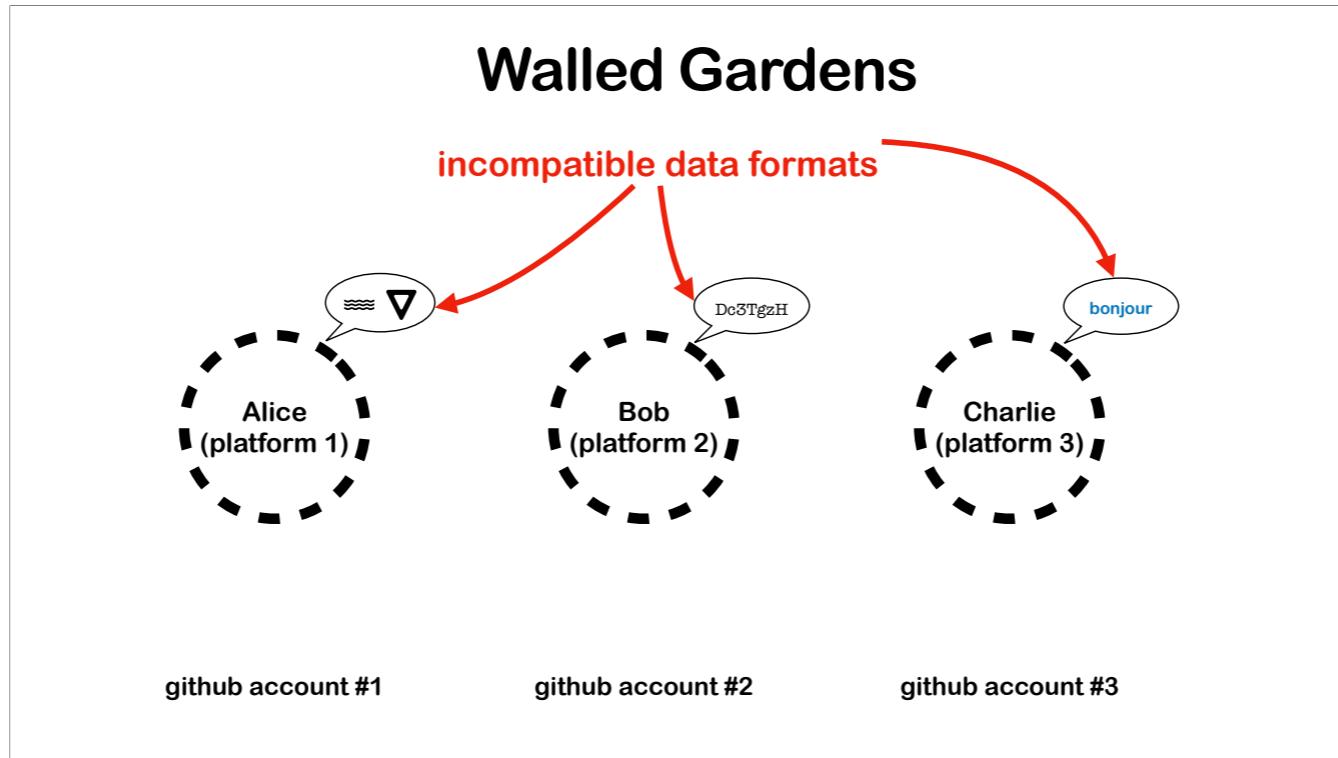
Moving to decentralized open source platforms help a little bit. Our data is at least a little more accessible. So I'll show the walls around these open source gardens as being broken down a bit; porous.

Walled Gardens

Open Source platforms: better, but they are still Walled Gardens. Why?



But as I see it, these are still essentially, walled gardens. Why is that? and what do I mean by that?



Consider the fact that open source projects, as they are implemented today, each uses its own distinct set of data formats. They are distinct, and therefore incompatible with each other. What do I mean by that?

These apps send information back and forth. Often, JSON files are what they use to send back and forth from one user to another.

```
{ "post": { "slug": "its-greats", "vendorID": { "peerID": "Qmc9mt2Ez8569sntWBjmX66MUFVHbpnH6vgGBCDJHxamTY", "pubkeys": { "identity": "CAESIAF+PySHGChTSeeIqMEL1T7AX3bRYCH1i3hxxgYo+RvJ", "bitcoin": "AoBy4q3qKb7WQQZVf8GNnmp1bDTKdWZSbUPkmfdavAG" }, "bitcoinSig": "MEUCIQCl2KP3Y7VDjVZs3EJunIDI13WC+916Z34NKoUr8Xqsw0IgDy8FMYAjh3FgUJXdMUCKBkLMLs4GgCfBDxcZgffjx+Y=" }, "status": " Its greats!!", "images": [ ], "tags": [ ], "postType": "COMMENT", "reference": "QmQUPt6ABdL3i5LvMJdYf7cq4gjT6ULFUSUtEkpNEAwj9/49cf7aa2-8efc-49fe-9388-526223fe7a0c", "timestamp": "2019-08-30T23:28:12.684431396Z" }, "signature": "NcbZF66Xf7y9fGQqFgXYnBgYLzWSVVmfMLG1+y834IjrzquNcR9DGdIo6g6STrEvvBZZV0fDxuwhUNbSxb1zDg==" } }
```

Here is an example of a JSON file which is a record of a post from OB1's Haven app, the decentralized Twitter. This is the JSON file that Alice's app would send from her Haven app on her smartphone over their forked version of the IPFS network to Bob's device. If Bob wants to check when this was posted,

```
{  
  "post": {  
    "slug": "its-greats",  
    "vendorID": {  
      "peerID": "Qmc9mt2Ez8569sntWBjmX66MUFVHbpnH6vgGBCDJHxamTY",  
      "pubkeys": {  
        "identity": "CAESIAF+PySHGChTSeeIqMEL1T7AX3bRYCH1i3hxxgYo+RvJ",  
        "bitcoin": "AoBy4q3qKb7W0QZVf8GNnmp1bDTKdWZSbUPkmfdavAG"  
      },  
      "bitcoinSig": "MEUCIQCl2KP3Y7VDjVZs3EJunIDI13WC+916Z34NKoUr8Xqsw0IgDy8FMYAjh3FgUJXdMUCKBkLMLs4GgCfBDxcZgffjx+Y=",  
    },  
    "status": " Its greats!!",  
    "images": [  
    ],  
    "tags": [  
    ],  
    "postType": "COMMENT",  
    "reference": "QmQUPt6ABdL3i5LvMJdYf7cq4gjT6ULFUSUtEkpNEAwj9/49cf7aa2-8efc-49fe-9388-526223fe7a0c",  
    "timestamp": "2019-08-30T23:28:12.684431396Z"  
  },  
  "signature": "NcbZF66Xf7y9fGQqFgXYnBgYLzWSVVmfMLG1+y834IjrzquNcR9DGdIo6g6STrEvvBZZV0fDxuwhUNbSxb1zDg=="  
}
```

his software would have to find this file; it would then look under the key called post, highlighted here in red;

```
{  
  "post": {  
    "slug": "its-greats",  
    "vendorID": {  
      "peerID": "Qmc9mt2Ez8569sntWBjmX66MUFVHbpnH6vgGBCDJHxamTY",  
      "pubkeys": {  
        "identity": "CAESIAF+PySHGChTSeeIqMEL1T7AX3bRYCH1i3hxxgYo+RvJ",  
        "bitcoin": "AoBy4q3qKb7WQQZVf8GNnmp1bDTKdWZSZ6UPkmfdavAG"  
      },  
      "bitcoinSig": "MEUCIQCl2KP3Y7VDjVZs3EJunIDI13WC+916Z34NKoUr8Xqsw0IgDy8FMYAjh3FgUJXdMUCKBkLMLs4GgCfBDxcZgffjx+Y=",  
    },  
    "status": " Its greats!!",  
    "images": [  
    ],  
    "tags": [  
    ],  
    "postType": "COMMENT",  
    "reference": "QmQUPt6ABdL3i5LvMJdYf7cq4gjT6ULFUSUtEkpNEAwj9/49cf7aa2-8efc-49fe-9388-526223fe7a0c",  
    "timestamp": "2019-08-30T23:28:12.684431396Z"  
  },  
  "signature": "NcbZF66Xf7y9fGQqFgXYnBgYLzWSVVmfMLG1+y834IjrzquNcR9DGdIo6g6STrEvvBZZV0fDxuwhUNbSxb1zDg=="  
}
```

then under timestamp.

```
{  
  "post": {  
    "slug": "its-greats",  
    "vendorID": {  
      "peerID": "Qmc9mt2Ez8569sntWBjmX66MUFVHbpnH6vgGBCDJHxamTY",  
      "pubkeys": {  
        "identity": "CAESIAF+PySHGChTSeeIqMEL1T7AX3bRYCH1i3hxxgYo+RvJ",  
        "bitcoin": "AoBy4q3qKb7W0QZv8GNnmp1bDTKdWZSbUPkmfdavAG"  
      },  
      "bitcoinSig": "MEUCIQCl2KP3Y7VDjVZs3EJunIDI13WC+916Z34NKoUr8Xqsw0IgDy8FMYAjh3FgUJXdMUCKBkLMLs4GgCfBDxcZgffjx+Y=",  
    },  
    "status": " Its greats!!",  
    "images": [  
    ],  
    "tags": [  
    ],  
    "postType": "COMMENT",  
    "reference": "0mQUPt6ABdL3i5LvMJdYf7cq4qjT6ULFUSUtEkpNEAwj9/49cf7aa2-8efc-49fe-9388-526223fe7a0c",  
    "timestamp": "2019-08-30T23:28:12.684431396Z"  
  },  
  "signature": "NcbZF66Xf7y9fGQqFgXYnBgYLzWSVVmfMLG1+y834IjrzquNcR9DGdIo6g6STrEvvBZZV0fDxuwhUNbSxb1zDg=="  
}
```

and here would be the information he was looking for.

```
{  
    "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQSTpnJYUsML7",  
    "handle": "",  
    "name": "OpenBazaar Store",  
    "location": "",  
    "about": "  
This is the official OpenBazaar store run by the development team.  
All sales from this store go into the OpenBazaar project fund.  
  
Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar  
",  
    "shortDescription": "The official OpenBazaar store run by the development team.",  
    "nsfw": false,  
    "vendor": true,  
    "moderator": false,  
    "contactInfo": {  
        "website": "openbazaar.org",  
        "email": "project@openbazaar.org",  
        "phoneNumber": "",  
        "social": [  
            {  
                "type": "Twitter",  
                "username": "openbazaar",  
                "proof": ""  
            }  
        ]  
    },  
    "colors": {  
        "primary": "#FFFFFF",  
        "secondary": "#ECEEF2",  
        "text": "#252525",  
        "highlight": "#2BAD23",  
        "highlightText": "#252525"  
    },  
    "avatarHashes": {  
        "tiny": "zb2rhclBhqq139K9r2DT1BPaqTqTVTsm0nNzdb1BZpdBK4Zk",  
        "small": "zb2rhWhoRxwekSNsrRdvFJhxTEzY5AHsdgXYh5bCpvGAD1o9",  
        "medium": "zb2rhg5hULqv465Chru7qKBvMfr81U3gsjTWo9Vd6NEA4D",  
        "large": "zb2rbhg95WEXEs4dohj9i3ZadqaTu4mGBShcVC1dP8SN9a",  
        "original": "zb2rhY5wuaETuNnMLVbG6BsGMTvcjb2jum6hYALSwW4C04E3w"  
    },  
    "headerHashes": {}  
}
```

As another example: Here is the file that would be used to convey information about one of the vendors. (The official OB store, in fact.)

```
{
  "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQSTpnJYUsML7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "",
  This is the official OpenBazaar store run by the development team.
  All sales from this store go into the OpenBazaar project fund.
  Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar
  ,
  "shortDescription": "The official OpenBazaar store run by the development team.",
  "nsfw": false,
  "vendor": true,
  "moderator": false,
  "contactInfo": {
    "website": "openbazaar.org",
    "email": "project@openbazaar.org",
    "phoneNumber": "",
    "social": [
      {
        "type": "Twitter",
        "username": "openbazaar",
        "proof": ""
      }
    ]
  },
  "colors": {
    "primary": "#FFFFFF",
    "secondary": "#ECEEF2",
    "text": "#252525",
    "highlight": "#2BAD23",
    "highlightText": "#252525"
  },
  "avatarHashes": {
    "tiny": "zb2rhclBhqq139K9r2DT1BPaqTqTVTSnm0nNzdb1BZpdBK4Zk",
    "small": "zb2rhWhoRxwekSNsrRdvFJhxTEzY5AHsdgXYh5bCpvGAD1o9",
    "medium": "zb2rhg5hULqv465Chru7qKBvMfr81U3gsjTWo9Vd6NEA4D",
    "large": "zb2rbqGW95WEXEs4dohj9i3ZadqaTu4mGBShcVC1dP8SN9a",
    "original": "zb2rhY5wuaETuNnMLVbG6BsGMTvcjb2jum6hYALSwW4C04E3w"
  },
  "headerHashes": {
}
```

If Alice wants to know the Twitter handle of this vendor, her app first of all, would have to find this file, so it would have to know where to look; it would then reach into this file

```
{  
    "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQSTpnJYUsML7",  
    "handle": "",  
    "name": "OpenBazaar Store",  
    "location": "",  
    "about": "  
This is the official OpenBazaar store run by the development team.  
All sales from this store go into the OpenBazaar project fund.  
  
Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar  
",  
    "shortDescription": "The official OpenBazaar store run by the development team.",  
    "nsfw": false,  
    "vendor": true,  
    "moderator": false,  
    "contactInfo": {  
        "website": "openbazaar.org",  
        "email": "project@openbazaar.org",  
        "phoneNumber": "",  
        "social": [  
            {  
                "type": "Twitter",  
                "username": "openbazaar",  
                "proof": ""  
            }  
        ]  
    },  
    "colors": {  
        "primary": "#FFFFFF",  
        "secondary": "#ECEEF2",  
        "text": "#252525",  
        "highlight": "#2BAD23",  
        "highlightText": "#252525"  
    },  
    "avatarHashes": {  
        "tiny": "zb2rhclbhqq139K9r2DT1BPaqTqTVTsm0nNzdb1BZpdBK4Zk",  
        "small": "zb2rhWhoRxwekSNsrRdvFJhxTEzY5AHsdgXYh5bCpvGAD1o9",  
        "medium": "zb2rhg5hULJqv465Chru7qKBvMfr81U3gsjTWo9Vd6NEA4D",  
        "large": "zb2rbqGW95WEXEs4dohj9i3ZadqaTu4mGBShcVC1dP8SN9a",  
        "original": "zb2rhY5wuaETuNnMLVbG6BsGMTvcjb2jum6hYALSwW4C04E3w"  
    },  
    "headerHashes": {}  
}
```

and look into contactInfo

```
{
  "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQSTpnJYUsML7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "",
  This is the official OpenBazaar store run by the development team.
  All sales from this store go into the OpenBazaar project fund.
  Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar
  ,
  "shortDescription": "The official OpenBazaar store run by the development team.",
  "nsfw": false,
  "vendor": true,
  "moderator": false,
  "contactInfo": {
    "website": "openbazaar.org",
    "email": "project@openbazaar.org",
    "phoneNumber": "",
    "social": [
      {
        "type": "Twitter",
        "username": "openbazaar",
        "proof": ""
      }
    ]
  },
  "colors": {
    "primary": "#FFFFFF",
    "secondary": "#ECEEF2",
    "text": "#252525",
    "highlight": "#2BAD23",
    "highlightText": "#252525"
  },
  "avatarHashes": {
    "tiny": "zb2rhclBhqq139K9r2DT1BPaqTqTVTsm0nNzdb1BZpdBK4Zk",
    "small": "zb2rhWhoRxwekSNsrRdvFJhxTEzY5AHsdgXYh5bCpvGAD1o9",
    "medium": "zb2rhg5hULqv465Chru7qKBvMfr81U3gsjTWo9Vd6NEA4D",
    "large": "zb2rbqGW95WEXEs4dohj9i3ZadqaTu4mGBShcVC1dP8SN9a",
    "original": "zb2rhY5wuaETuNnMLvb6BsGMTvcjb2jum6hYALSwW4C04E3w"
  },
  "headerHashes": {
}
```

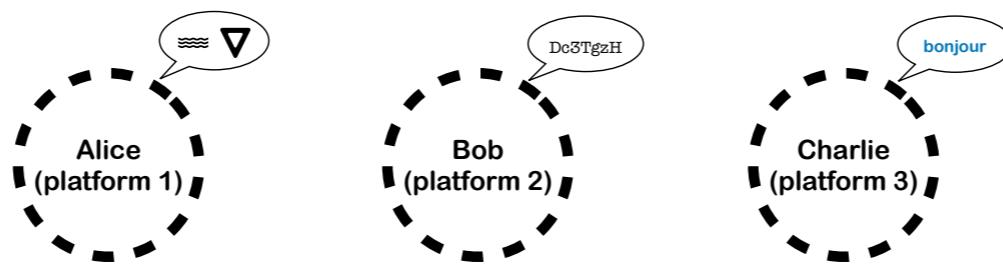
then under that into social, which the software would have to know is an array, and it would have to know that the elements of that array are objects, and so it would look through each one of those objects and find one
that has key equal to type; and value equal to Twitter, and in that object,

```
{
  "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQSTpnJYUsML7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "",
  This is the official OpenBazaar store run by the development team.
  All sales from this store go into the OpenBazaar project fund.
  Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar
  ,
  "shortDescription": "The official OpenBazaar store run by the development team.",
  "nsfw": false,
  "vendor": true,
  "moderator": false,
  "contactInfo": {
    "website": "openbazaar.org",
    "email": "project@openbazaar.org",
    "phoneNumber": "",
    "social": [
      {
        "type": "Twitter",
        "username": "openbazaar",
        "proof": ""
      }
    ]
  },
  "colors": {
    "primary": "#FFFFFF",
    "secondary": "#ECEEF2",
    "text": "#252525",
    "highlight": "#2BAD23",
    "highlightText": "#252525"
  },
  "avatarHashes": {
    "tiny": "zb2rhclBhqq139K9r2DT1BPaqTqTVTsm0nNzdb1BZpdBK4Zk",
    "small": "zb2rhWhoRxwekSNsrRdvFJhxTEzY5AHsdgXYh5bCpvGAD1o9",
    "medium": "zb2rhg5hULqv465Chru7qKBvMfr81U3gsjTWo9Vd6NEA4D",
    "large": "zb2rbqGW95WEXEs4dohj9i3ZadqaTu4mGBShcVC1dP8SN9a",
    "original": "zb2rhY5wuaETuNnMLVbG6BsGMTvcjb2jum6hYALSwW4C04E3w"
  },
  "headerHashes": {
}
```

it would look for the key: username. and her app would be able to tell her that the Twitter username for this vendor is openbazaar. Her app won't be able to do all of that unless it already knows how files of this type are to be formatted. Which as you can see, can be kinda tedious.

Walled Gardens

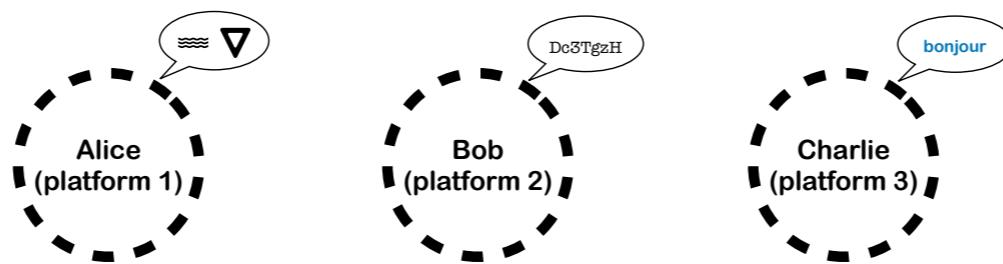
Different platforms -> different data formats -> different languages -> Walled Gardens



If two users are not on the same platform, then even if their apps have the same or similar data, they are not going to be using the same formats. Not using the same formats is like using different languages. And that leads to Walled Gardens.

Walled Gardens

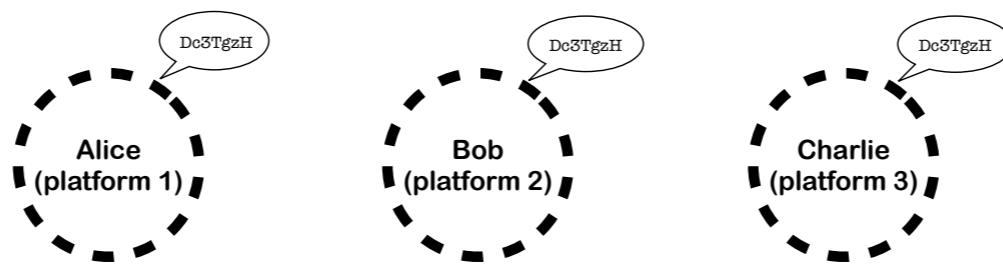
OK Solution: build translators, or ...



What are the solutions to different languages? Well one might be to build translators,

Walled Gardens

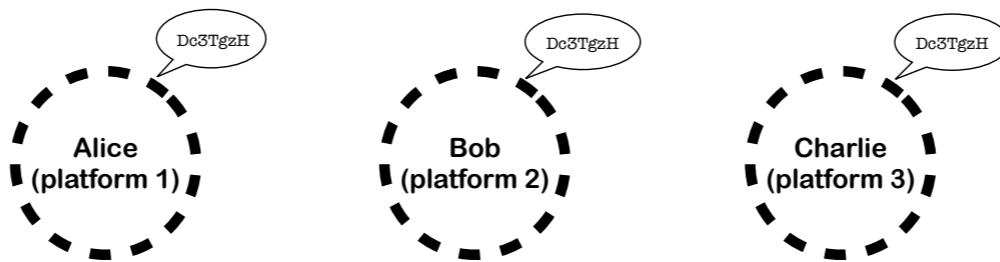
Better Solution: speak the same language!



But a better solution would be to get them (metaphorically) to speak the same language!

Walled Gardens

Better Solution: speak the same language!
(shared data formats)



Get them to agree on data formats; not necessarily for everything, but at least for whatever data they want to share. That's a better solution, I think. And this is the reason the CG and Grapevine exist.

[end 07]

8

What does “Data Format” mean?

[start 08]

Let me take a step back and look at what I mean by Data Format.

What does “Data Format” mean?

file extension

.png, .jpg, ...

.csv, .json, .xml, ...

.exe

The first thing that may come to mind when you hear data format might be: FILE EXTENSION. jpeg, CSV, etc.
But when I say file format, I mean formatting details that extend beyond simply the file extension.

What does “Data Format” mean?

file extension

.png, .jpg, ...

.csv, .**json**, .xml, ...

.exe

I will use JSON files to illustrate what I mean. J-S-O-N stands for javascript object notation. Objects in general, and JSON in particular, are widely used for many applications, and this is the format that I am focusing on the most in this presentation. (Although I will note, as an aside, that the thought process that I am following in this presentation could be applied just as well to some different file format, other than JSON; like XML, or SQL, or anything.)

What does “Data Format” mean?

file extension

- .png, .jpg, ...
- .csv, .json, .xml, ...
- .exe

userInfo

```
{  
  "name": "Alice",  
  "hometown": "Nashville"  
}
```

What does “Data Format” mean?

file extension

- .png, .jpg, ...
- .csv, .json, .xml, ...
- .exe

userInfo

```
{           {  
  "name": "Alice",   "userData": {  
  "hometown": "Nashville"    "username": "Bob",  
}           "city": "New York"  
},  
          "metadata": {  
}           "lastUpdate": 1640984720  
}  
}
```

Here are two different sample JSON files, each one of which might be an example of a file with user information in it.

What does “Data Format” mean?

file extension

- .png, .jpg, ...
- .csv, .json, .xml, ...
- .exe

userInfo

```
{           {  
  "name": "Alice",   "userData": {  
  "hometown": "Nashville"    "username": "Bob",  
}           "city": "New York"  
},  
          "metadata": {  
}           "lastUpdate": 1640984720  
}  
}
```

When I say data format, in the context of JSON files, I am referring to the parts in red. A file that specifies how the parts in red need to be structured, is sometimes called a JSON Schema.

What does “Data Format” mean?

file extension

.png, .jpg, ...

.csv, .json, .xml, ...

.exe

userInfo

```
{ "name": "Alice",  
  "hometown": "Anytown"  
}  
  
{  
  "userData": {  
    "username": "Bob",  
    "city": "New York"  
  },  
  "metadata": {  
    "lastUpdate": 1640984720  
  }  
}
```

JSON Schema

As you can see, the JSON Schema for the file on the left is different than the JSON Schema for the file on the right; which is another way of saying that these two files are formatted differently.

JSON file

```
{  
  name: Alice,  
  hometown: Anytown  
}
```

To explain JSON Schema, let me explain a little about JSON files. Essentially a JSON file is a collection of things that are called properties.

JSON file

```
{  
  name: Alice,  
  hometown: Anytown  
}
```

property

```
key: name  
type: string
```

A property has a key and a value. An example is here, highlighted in red. The property key is always a string: here, the key is name. The property value can be a string, as it is in this case.

JSON file

```
{  
    name: Alice,  
    hometown: Anytown  
}
```

property

key: hometown
type: string

Here is a second property, with key: hometown, the value of which is also a string.

JSON file	JSON Schema
{ name: Alice, hometown: Anytown }	

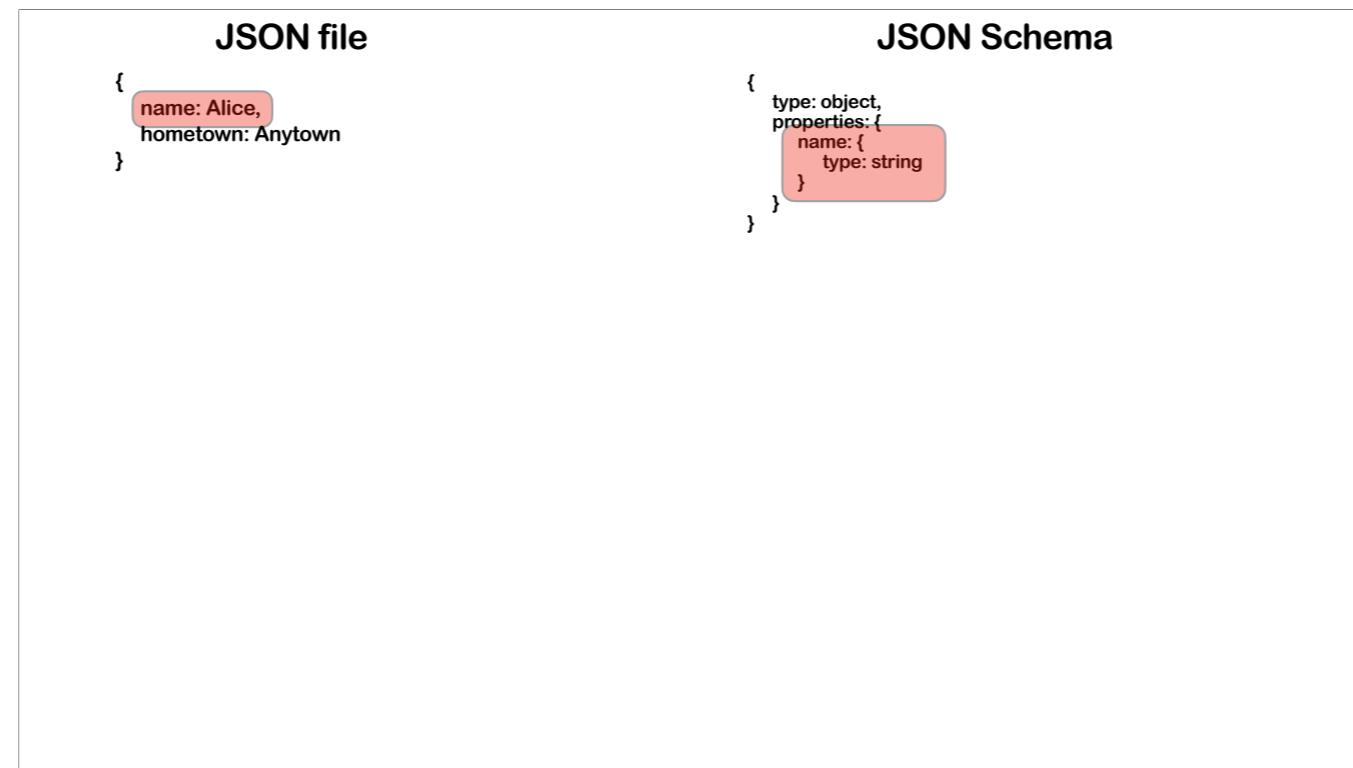
The format of this JSON file can be recorded within a JSON Schema, which is itself, also a JSON file.

JSON file	JSON Schema
{ name: Alice, hometown: Anytown }	{ type: object, properties: { } }

I'll show you how the format of the properties in the file on the left are recorded in the JSON Schema on the right. First, we initialize the JSON Schema, as you see here.

JSON file	JSON Schema
{ name: Alice, hometown: Anytown }	{ type: object, properties: { } }

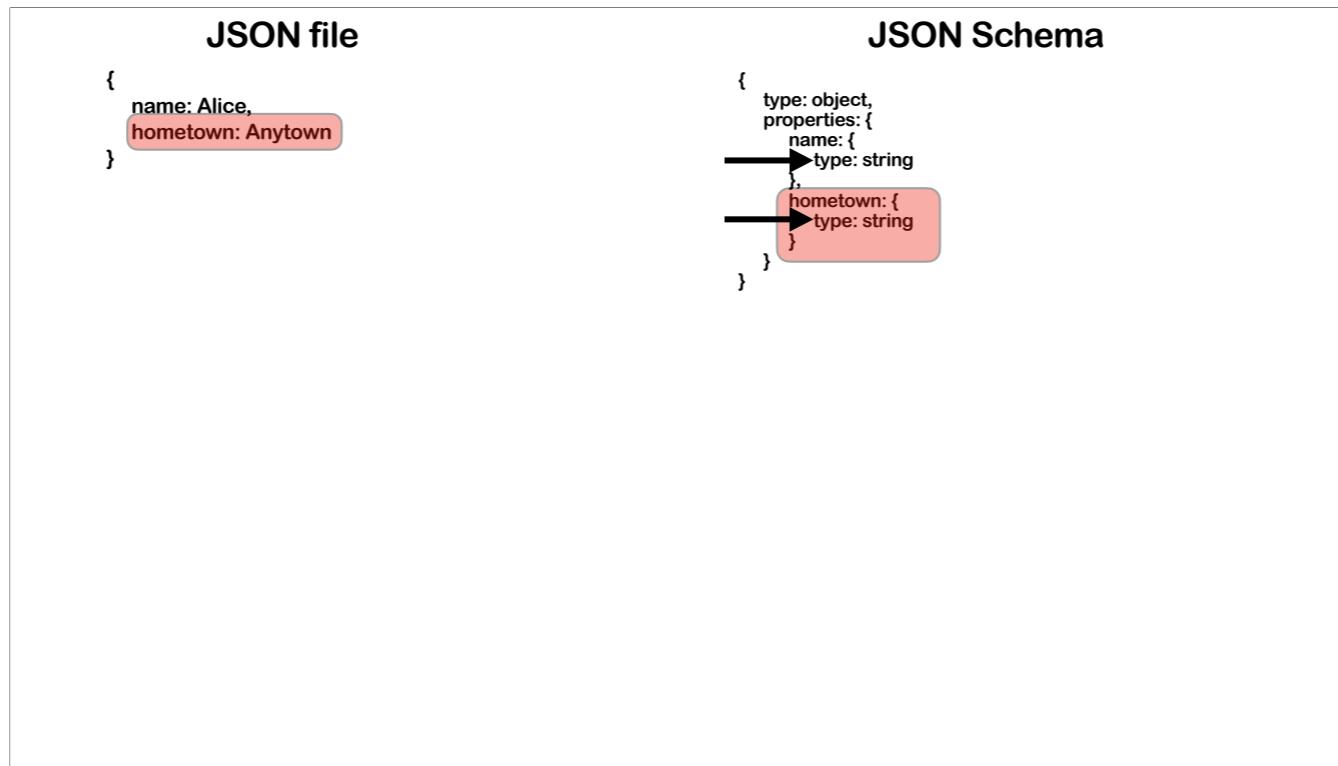
The first property, with key: name,



is recorded on the right as you see here.

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string } } }</pre>

The second property, with key: hometown, is likewise recorded on the right as you see here.



The values for both of these properties can be anything as long as they are of type: string.

JSON file

```
{  
  name: Alice,  
  hometown: Anytown  
}
```

JSON Schema

```
{  
  type: object,  
  properties: {  
    name: {  
      type: string  
    },  
    hometown: {  
      type: string  
    }  
  }  
}
```

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27 }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer } } }</pre>

Type can also be an integer, as you see with the next property, with key: age.

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean } } }</pre>

The type can be a boolean, as is the case with the next property, with key: married.

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false, siblings: [Bob, Charlie] }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean }, siblings: { type: array, items: { type: string } } } }</pre>

The value can be of type: array. In this example, the items of the array are strings.

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false, siblings: [Bob, Charlie], contactInfo: { email: alice@herdomain.com, address: 123 Main Street } }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean }, siblings: { type: array, items: { type: string } }, contactInfo: { type: object, properties: { email: { type: string }, address: { type: string } } } } }</pre>

The most interesting scenario is when the property type is what is called an object. That means that the property is essentially a collection of more properties. In this example, the property with key: contactInfo, is composed of two sub-properties:

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false, siblings: [Bob, Charlie], contactInfo: { email: alice@herdomain.com, address: 123 Main Street } }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean }, siblings: { type: array, items: { type: string } }, contactInfo: { type: object, properties: { email: { type: string }, address: { type: string } } } } }</pre>

the first one being email,

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false, siblings: [Bob, Charlie], contactInfo: { email: alice@herdomain.com, address: 123 Main Street } }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean }, siblings: { type: array, items: { type: string } }, contactInfo: { type: object, properties: { email: { type: string }, address: { type: string } } } } }</pre>

the second one being: address.

JSON file	JSON Schema
<pre>{ name: Alice, hometown: Anytown, age: 27, married: false, siblings: [Bob, Charlie], contactInfo: { email: alice@herdomain.com, address: 123 Main Street } }</pre>	<pre>{ type: object, properties: { name: { type: string }, hometown: { type: string }, age: { type: integer }, married: { type: boolean }, siblings: { type: array, items: { type: string } }, contactInfo: { type: object, properties: { email: { type: string }, address: { type: string } } } } }</pre>

As you can see, keeping track of the JSON file's format, in this case using a JSON Schema as the record, is no simple matter. And the number of different file formats, is effectively, unlimited.

[end 08]

9

There are many ways to format data

formatting options

analog digital

[start 09]

Let's back up and take more of a bird's eye view. Suppose you want to store some data. You need to pick a format. Well, first you decide: analog or digital.

There are many ways to format data

formatting options

analog

digital

Let's say, digital.

There are many ways to format data

formatting options

analog

digital



But that's only the first in a series of choices. Think of it like a drill down menu.

There are many ways to format data

formatting options

analog

digital

ternary

binary



The next choice could be: ternary or binary.

There are many ways to format data

formatting options

analog digital
ternary binary

Everyone uses binary, so let's go with binary.

There are many ways to format data

formatting options

analog digital

ternary binary

7 8 9

Next: is each byte 7, 8, or 9 bits.

There are many ways to format data

formatting options

analog digital

ternary binary

7 8 9

Let's go with 8. That's what everyone else is using, so we'll just go with that.

There are many ways to format data

formatting options

analog digital
ternary binary
7 8 9
sql json xml jpeg

Now we have to pick a file type.

There are many ways to format data

formatting options

analog digital
ternary binary
7 8 9
sql json xml jpeg

Let's go with JSON.

There are many ways to format data

formatting options	# of readily available options
analog	digital
ternary	binary
7	8
sql	json
	xml
	jpeg

So far, the number of readily available options, meaning options that are in wide usage,

There are many ways to format data

formatting options	# of readily available options
analog	digital
ternary	binary
7	8
sql	json
	xml
	jpeg

limited

has been limited. And depending on the context, the best solution may be obvious, meaning the number of reasonable options for some of these steps, is one.

There are many ways to format data

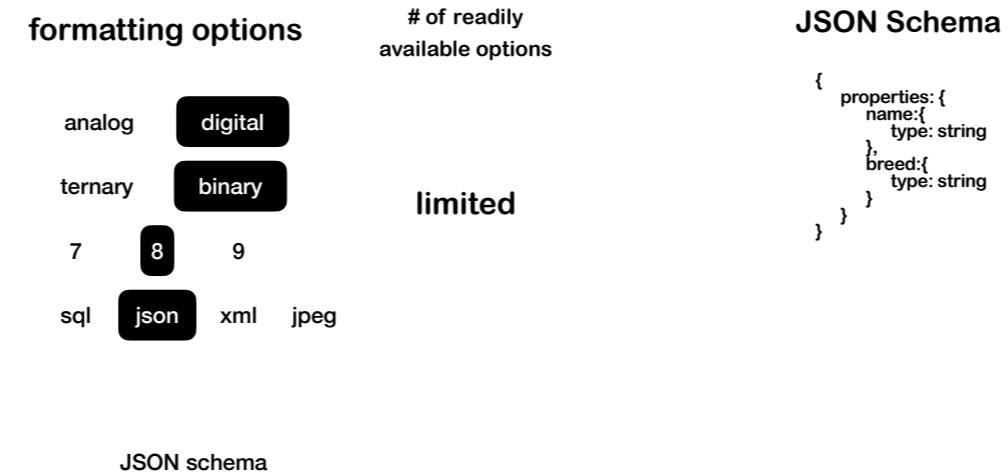
formatting options	# of readily available options
analog	digital
ternary	binary
7	8
sql	json
	xml jpeg

limited

JSON schema

But let's keep drilling down further. The next thing we have to decide, for a JSON file, is the JSON Schema.

There are many ways to format data



As an example, here would be the JSON Schema

There are many ways to format data

formatting options

analog digital
ternary binary
7 8 9
sql json xml jpeg

of readily available options

limited

JSON Schema

```
{  
  properties: {  
    name:{  
      type: string  
    },  
    breed:{  
      type: string  
    }  
}
```

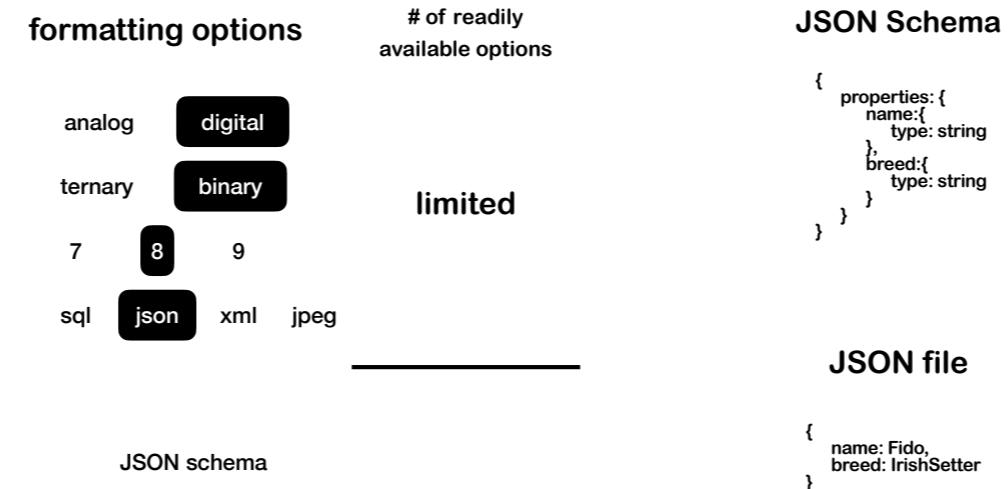
JSON schema

JSON file

```
{  
  name: Fido,  
  breed: IrishSetter  
}
```

which would describe the format of a JSON file such as this one.

There are many ways to format data



but unlike the number of readily available options so far, the number of ways to structure a JSON Schema

There are many ways to format data

formatting options

analog digital
ternary binary
7 8 9
sql json xml jpeg

of readily available options

limited

JSON Schema

```
{  
  properties: {  
    name:{  
      type: string  
    },  
    breed:{  
      type: string  
    }  
  }  
}
```

JSON schema

unlimited

JSON file

```
{  
  name: Fido,  
  breed: IrishSetter  
}
```

is unlimited.

There are many ways to format data

formatting options	# of readily available options		
analog	digital		
ternary	binary		
7	8		
9			
sql	json	xml	jpeg

limited

And this is not true only for JSON files. It is true for other types of data files as well.

There are many ways to format data

formatting options	# of readily available options
analog	digital
ternary	binary
7	8
9	
sql	json xml jpeg

limited

Suppose that instead of JSON file, we decided to store data in a relational database. That means we might have an SQL file that stores our data.

There are many ways to format data

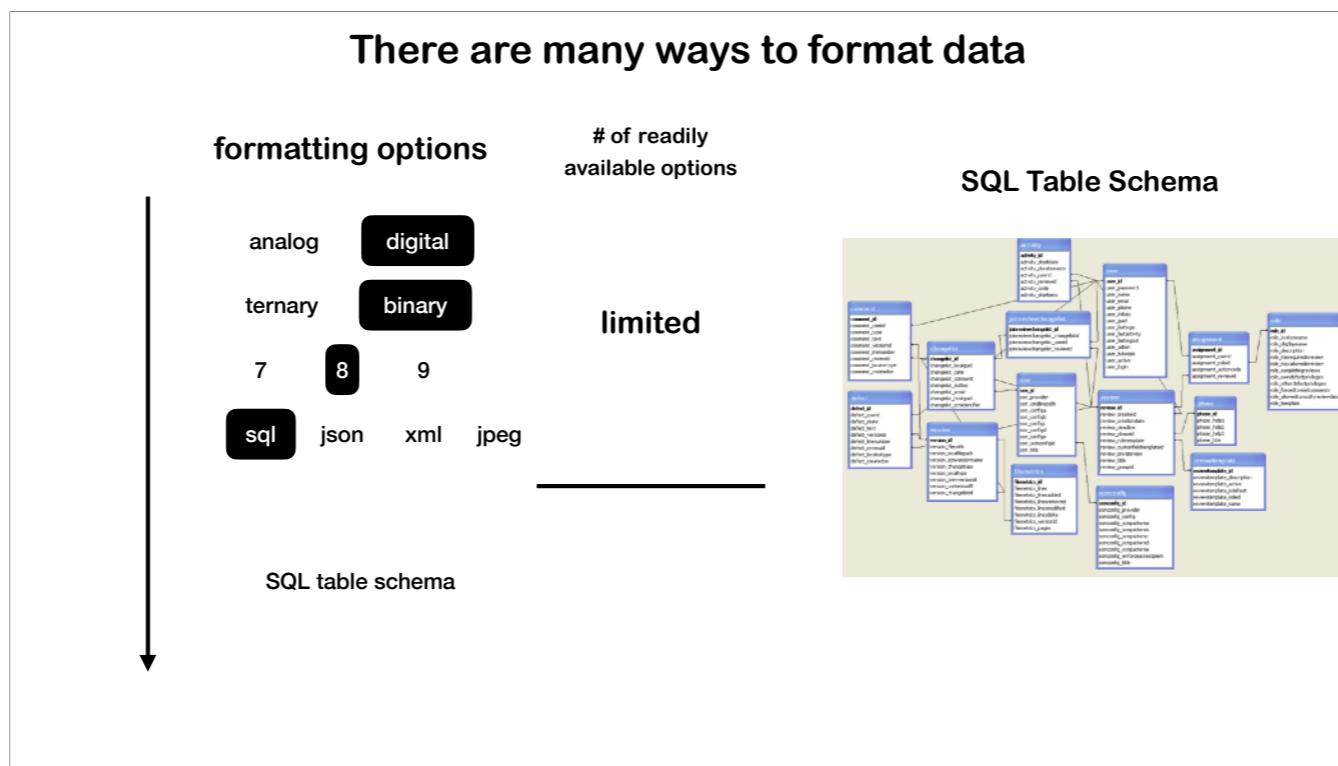
formatting options	# of readily available options
analog	digital
ternary	binary
7	8
sql	json xml jpeg

limited

SQL table schema

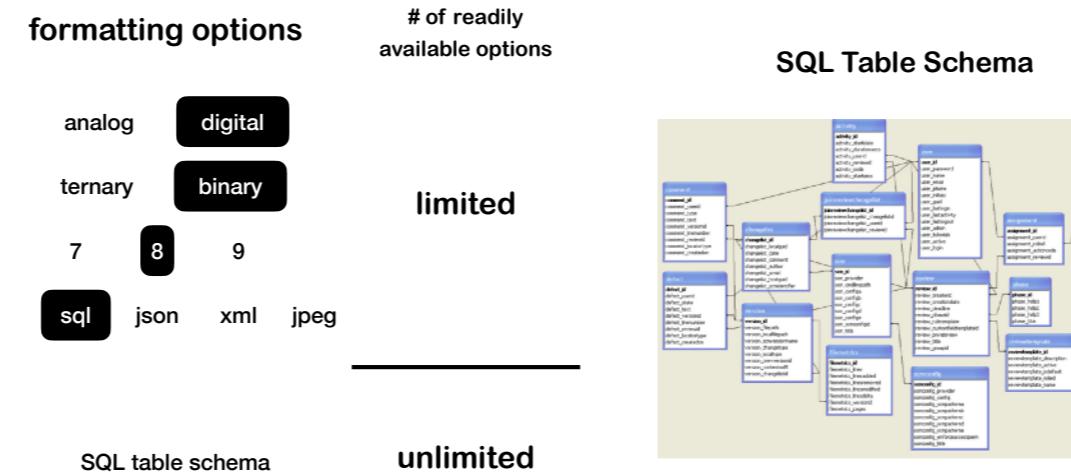
Just like JSON files have JSON Schema as a record of their format, so too SQL data is arranged according to a SQL table schema;

There are many ways to format data



in this example on the right, the SQL Table Schema is composed of 15 tables; each table has a name, with multiple columns, each with its own data type. How many ways are there to construct an SQL table schema?

There are many ways to format data



Unlimited. Infinite.

The number of ways to format data is

UNLIMITED

The number of ways to format data, once we drill down far enough, is UNLIMITED.

The number of ways to format data is

UNLIMITED

The number of ways to format data is

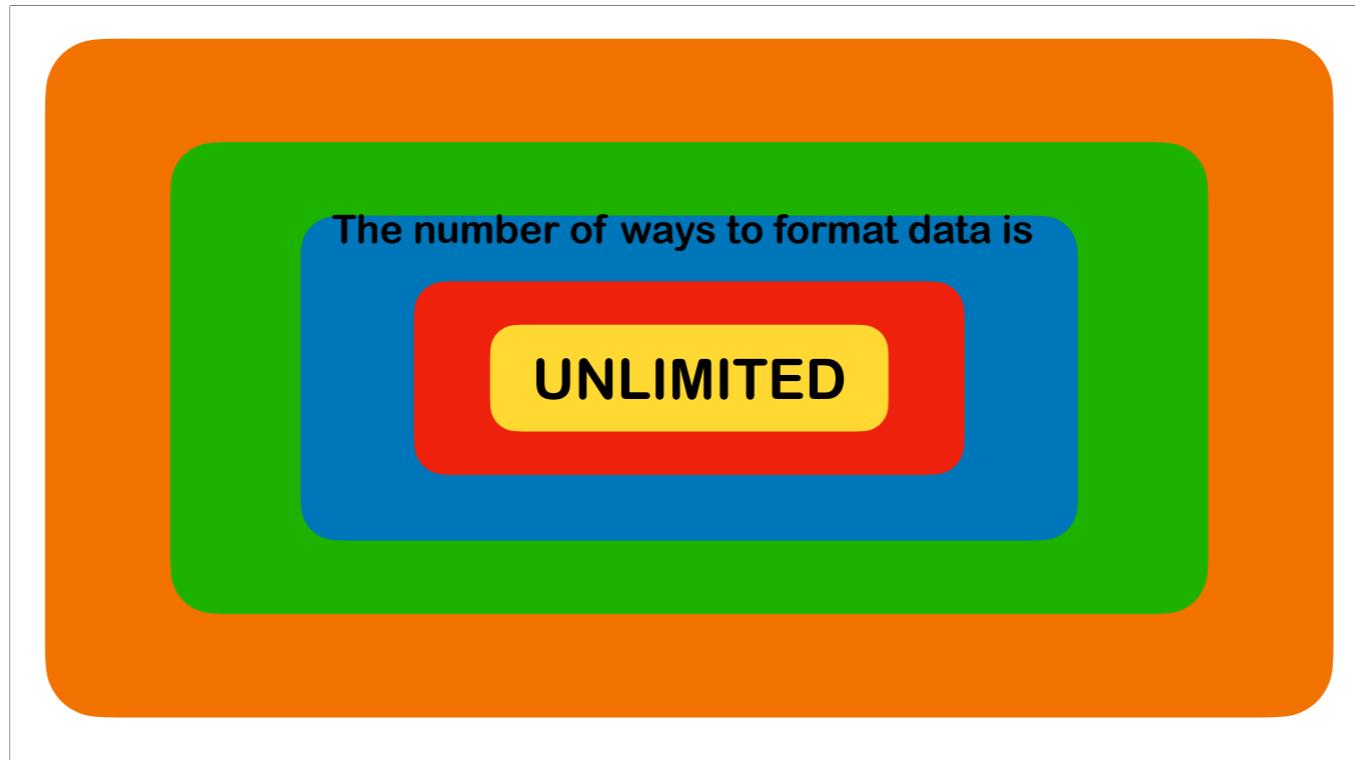
UNLIMITED

The number of ways to format data is

UNLIMITED

The number of ways to format data is

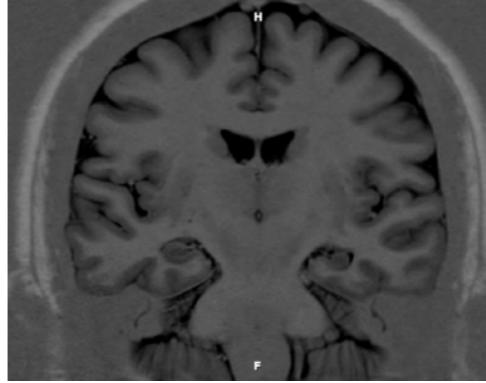
UNLIMITED



And that's an important observation.

```
{
  "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQ5TpJYUsML7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "This is the official OpenBazaar store run by the development team.  
All sales from this store go into the OpenBazaar project fund.  
Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar
",
  "shortDescription": "The official OpenBazaar store run by the development team",
  "nsfw": false,
  "vendor": true,
  "moderator": false,
  "contactInfo": {
    "website": "openbazaar.org",
    "email": "project@openbazaar.org",
    "phoneNumber": "",
    "social": [
      {
        "type": "Twitter",
        "username": "openbazaar",
        "proof": ""
      }
    ]
  },
  "colors": {
    "primary": "#FFFFFF",
    "secondary": "#ECEEF2",
    "text": "#252525",
    "highlight": "#28AD23",
    "highlightText": "#252525"
  },
  "avatarHashes": {
    "tiny": "zb2rhcbLbHqq139K9r2DT1BPaqTdTVTSnmQnNzdb1BZpdBk4Zk",
    "small": "zb2rbhwnoRxwekSNsrrDvFJHxTEzY5AHsdgXYh5bPCpVGAD1o9",
    "medium": "zb2rhg5hDULJqv465ChRu7qKBvMfr81U3gsjTWo9Vd6yNEA4D",
    "large": "zb2rbhbqGW95WEExE4D0H9i3ZAdqaTu4mGVBShcVC1dP8SN9a",
    "original": "zb2rhY5wuETuNhMLjB6GBsGMTvcjb2jum6hYALSwW4CQ4E3w"
  },
  "headerHashes": {
    "tiny": "zb2rhmxsgh9ehHtuJGNmTwc5gi1ogL3f0uZkJ6isa7kXPeWoJ",
    "small": "zb2rhesRQSxbHM4QhpwqcmXwHTy1JbgWGefxm5FK4CaV8p1N"
  }
}
```

The number of ways to format data is UNLIMITED



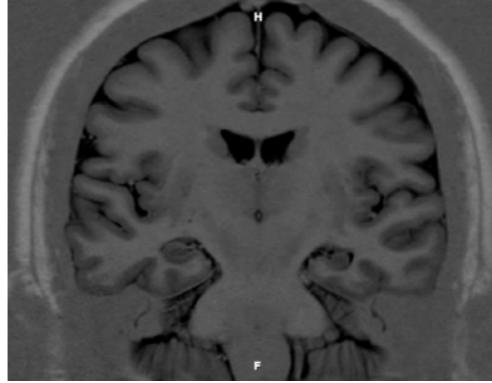
This is true for a data file. And I will conjecture, that this is also true in the cerebral cortex, I conjecture, for a cortical column or for whatever structure in the cortex is the analogue of a data file, assuming there is such a structure for which it would be worthwhile to make an analogy.

```
{
  "peerID": "QmcUDmZK8PsPYWw5FRHKNZFjszm2K6e68BQ5TpJYUsML7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "",
  This is the official OpenBazaar store run by the development team.
  All sales from this store go into the OpenBazaar project fund.

  Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar

  ,
  "shortDescription": "The official OpenBazaar store run by the development team",
  "nsfw": false,
  "vendor": true,
  "moderator": false,
  "contactInfo": {
    "website": "openbazaar.org",
    "email": "project@openbazaar.org",
    "phoneNumber": "",
    "social": [
      {
        "type": "Twitter",
        "username": "openbazaar",
        "proof": ""
      }
    ]
  },
  "colors": {
    "primary": "#FFFFFF",
    "secondary": "#ECEEF2",
    "text": "#252525",
    "highlight": "#28AD23",
    "highlightText": "#252525"
  },
  "avatarHashes": {
    "tiny": "zb2rhcbLbHqq139K9r2DT1BPaqTdTVTSnmQnNzdb1BZpdk4Zk",
    "small": "zb2rbhwnoRxwekSNsrrDvFJHxTEzY5AHsgdXYh5bPCpVGAD1o9",
    "medium": "zb2rhg5hDULJqv465ChRu7qKBvMfr81U3gsjTWo9Vd6yNEA4D",
    "large": "zb2rbhbqGW95WEExE4D0H9i3ZAdqaTu4mGBvShcVC1dP8SN9a",
    "original": "zb2rhYSwuaETuNhMLjB6GBsGMTvcjb2jum6hYALSwW4CQ4E3w"
  },
  "headerHashes": {
    "tiny": "zb2rhmxsgh9ehHtuJGNmTwc5gi1ogL3f0uZkJ6isa7kXPeWoj",
    "small": "zb2rhesRQSxbHM4QhpwqcmXwHTy1JbqWGefxmj5FK4CaV8p1N"
  }
}
```

The number of ways to format data is UNLIMITED



With the set of all possible data formats being unlimited: for any given data file, it stands to reason that the file format must be recorded somewhere. Where? The PoL says that it is somewhere nearby. Somewhere ACCESSIBLE. Easily Locatable. And that's it. We ought to implement this principle for the dWeb; maybe, it is ALREADY an undiscovered but essential feature of the structural organization of the cortex.

The number of ways to format data is

UNLIMITED

Hard to imagine that that would not be true. If it's not true, how could the system function? Guess what? We don't have a functioning dWeb. Not really. Not living up to its fullest potential. Not accomplishing what we all know, deep down, that it CAN accomplish, if only we can figure out how. I know that bc FaceBook still exists. Google still exists. Implementing the PoL is an ABSOLUTELY NECESSARY PREREQUISITE for changing that.

```
{ "post": { "slug": "its-greats", "vendorID": { "peerID": "Qmc9mt2Ez8569sntWBjmX66MUFVHbpnH6vgGBCDJHxamTY", "pubkeys": { "identity": "CAESIAF+PySHGChTSeeIqMEL1T7AX3bRYCH1i3hxxgYo+RvJ", "bitcoin": "AoBy4q3qKb7WQQZVf8GNnmp1bDTKdWZSZ6UPkmfdavAG" }, "bitcoinSig": "MEUCIQCl2KP3Y7VDjVZs3EJunIDI13WC+916Z34NKoUr8Xqsw0IgDy8FMYAjh3FgUJXdMUCKBkLMLs4GgCfBDxcZgffjx+Y=" }, "status": " Its greats!!", "images": [ ], "tags": [ ], "postType": "COMMENT", "reference": "QmQUPt6ABdL3i5LvMJdYf7cq4gjT6ULFUSUtEkpNEAwj9/49cf7aa2-8efc-49fe-9388-526223fe7a0c", "timestamp": "2019-08-30T23:28:12.684431396Z" }, "signature": "NcbZF66Xf7y9fGQqFgXYnBgYLzWSVVmfMLG1+y834IjrzquNcR9DGdIo6g6STrEvvBZZV0fDxuwhUNbSxb1zDg==" }}
```

Consider the question: where is the JSON Schema stored of a data file for a decentralized app like Open Bazaar or Haven? Where can you find the JSON Schema for a file like this one? In theory, it might be in documentation somewhere, at the project website, or at their GitHub repository. In practice, it is hard to find. Any documentation is probably incomplete. Probably not kept up to date. In practice, the JSON Schema for this file probably exists in the minds of God and the developer who wrote this code, and nowhere else. Which means that the only way for someone else to figure it out, would be to reverse engineer the whole thing. Which, I can tell you from experience, is a pain in the ass.

**The number of ways to format data is
UNLIMITED**

If Alice and Bob are using two distinct open source platforms, then they are relying on two distinct dev teams, and all of their data is formatted differently. And to make matters worse, a record, a detailed explanation, of those data formats are probably in the heads of some devs, and nowhere else. And that is why they can't communicate. That's why there are still living in walled gardens.

[end 09]

10

Communication requires pre-established mutual understanding, i.e. consensus on the relevant data format(s)

- so Alice can ask a coherent question / request data
- so Bob can answer / respond coherently
- so Alice can interpret his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

[start 10]

Suppose Alice and Bob want to communicate; but suppose they are NOT on the same platform. The whole point of the dWeb is that they should be able to do things like: curate a Twitter feed, or curate ratings of a product or business; and to do so in a meaningful way; but WITHOUT being tied to any platform, whether proprietary or open source.

Communication requires pre-established mutual understanding, i.e. consensus on the relevant data format(s)

- so Alice can ask a coherent question / request data
- so Bob can answer / respond coherently
- so Alice can interpret his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

What do I mean: communicate? Well we can divide it into 3 steps.

Communication requires pre-established mutual understanding, i.e. consensus on the relevant data format(s)

- Step 1**
- so Alice can ask a coherent question / request data
 - so Bob can answer / respond coherently
 - so Alice can interpret his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

Step 1: Alice requests through her app some piece of data.

Communication requires pre-established mutual understanding, i.e. consensus on the relevant data format(s)

- Step 1 • so Alice can ask a coherent question / request data
- Step 2 • so Bob can answer / respond coherently
• so Alice can interpret his response

Alice
(Platform 1)

Bob
(Platform 2)

Step 2: Bob responds.

Communication requires pre-established mutual understanding, i.e. consensus on the relevant data format(s)

- Step 1** • so Alice can ask a coherent question / request data
- Step 2** • so Bob can answer / respond coherently
- Step 3** • so Alice can interpret his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

Step 3: Alice interprets his response.

Communication requires pre-established mutual understanding, i.e. **consensus on the relevant data format(s)**

- Step 1** • so Alice can ask a coherent question / request data
- Step 2** • so Bob can answer / respond coherently
- Step 3** • so Alice can interpret his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

If Alice and Bob have pre-established consensus on the relevant data formats,

Communication requires pre-established mutual understanding, i.e. **consensus on the relevant data format(s)**

- Step 1** • so Alice can **ask** a coherent question / request data
- Step 2** • so Bob can **answer** / respond coherently
- Step 3** • so Alice can **interpret** his response

**Alice
(Platform 1)**

**Bob
(Platform 2)**

then all 3 steps can happen smoothly, effectively, meaningfully, and easily.

Note that it is not necessary for Alice and Bob to have pre-established consensus on ALL data formats. Only on the ones that are relevant to whatever conversation or interaction Alice and Bob are interested in having at the time.

Communication requires pre-established mutual understanding, i.e. **consensus on the relevant data format(s)**

- Step 1 • so Alice can **ask** a coherent question / request data
- Step 2 • so Bob can **answer** / respond coherently
- Step 3 • so Alice can **interpret** his response

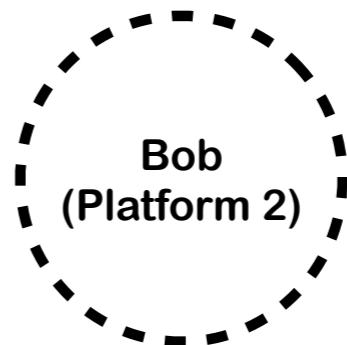
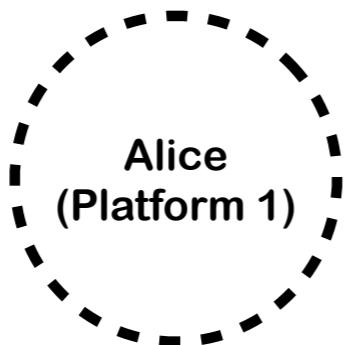
The **absence of such consensus leads to walled gardens!**

**Alice
(Platform 1)**

**Bob
(Platform 2)**

But in the absence of such consensus, none of those three steps can take place.

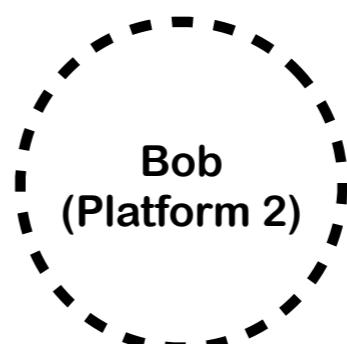
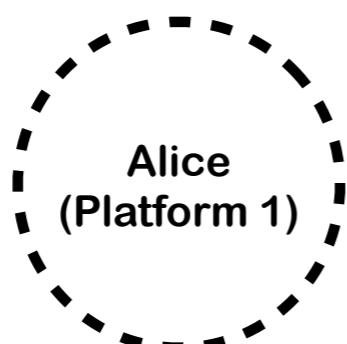
Data Formats and Walled Gardens



and we have Walled Gardens.

Data Formats and Walled Gardens

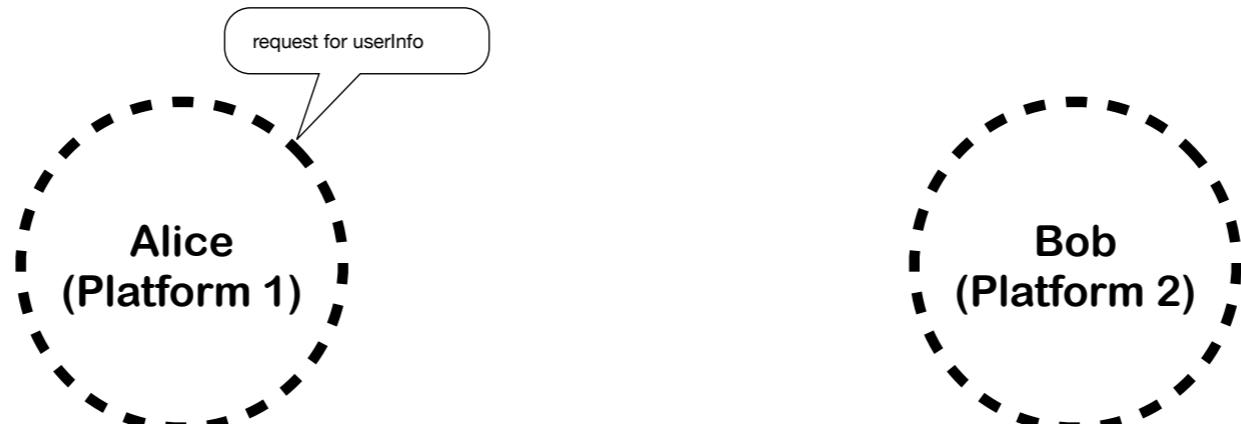
“Tell me about yourself.”



For example: if Alice wants to ask “tell me about yourself.”

Data Formats and Walled Gardens

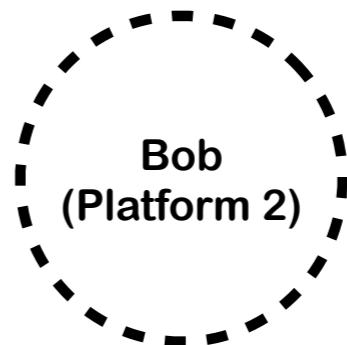
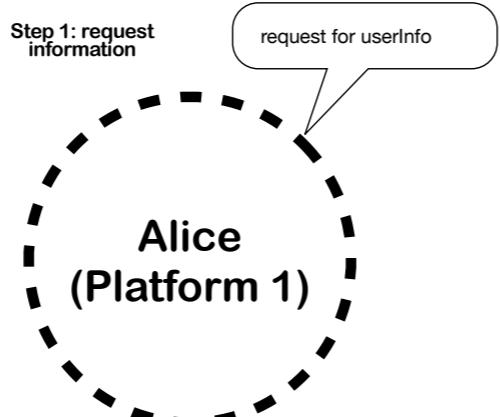
“Tell me about yourself.”



Supposing they had a pre-established mutual understanding or consensus about the relevant data formats, then whatever software she is using might send a request something like this: Please provide userInfo about yourself.

Data Formats and Walled Gardens

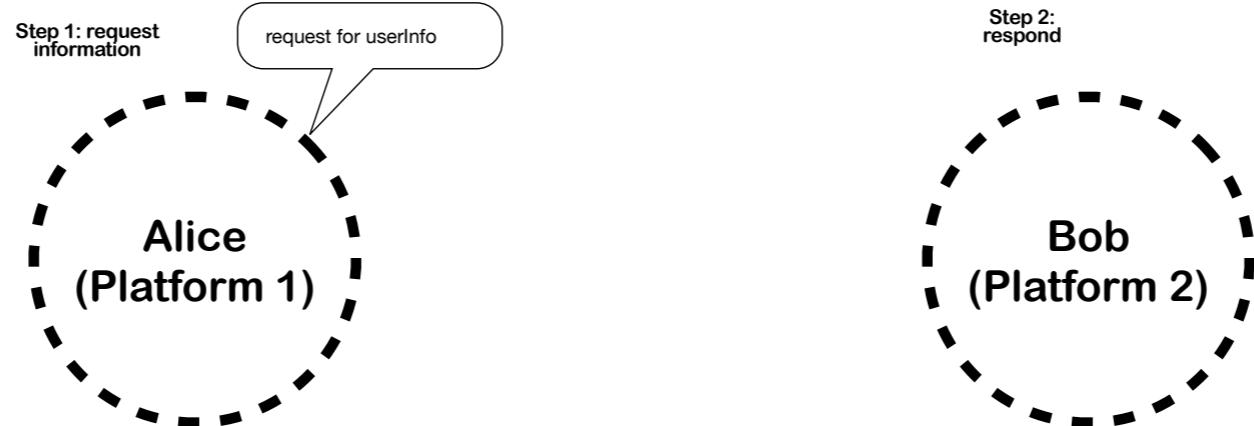
“Tell me about yourself.”



That corresponds to step 1: ask a coherent question.

Data Formats and Walled Gardens

“Tell me about yourself.”



Step 2 would be that Bob's software responds,

Data Formats and Walled Gardens

“Tell me about yourself.”

Step 1: request
information

request for userInfo

Alice
(Platform 1)

Step 2:
respond

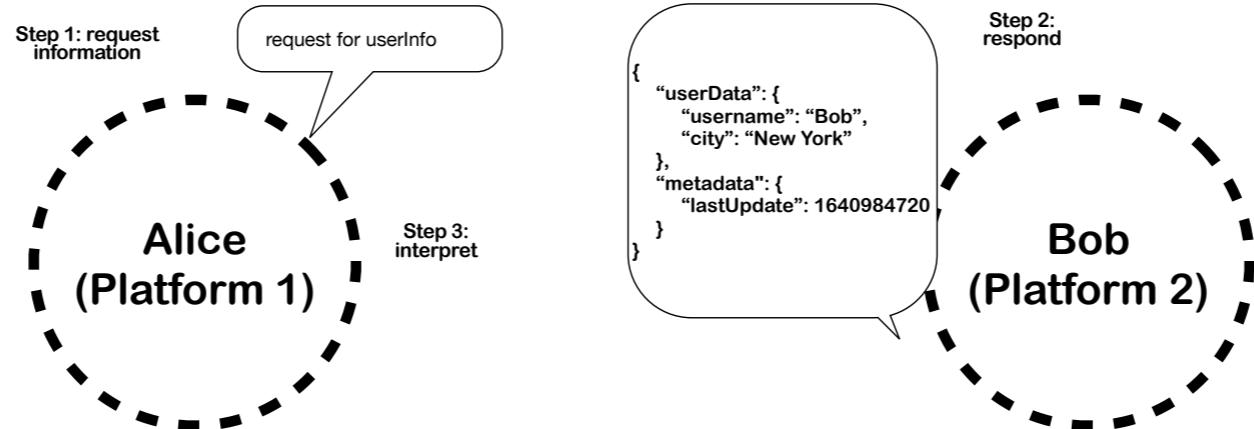
```
{  
  "userData": {  
    "username": "Bob",  
    "city": "New York"  
  },  
  "metadata": {  
    "lastUpdate": 1640984720  
  }  
}
```

Bob
(Platform 2)

with a file like this one.

Data Formats and Walled Gardens

“Tell me about yourself.”



And step 3: Alice’s software interprets the response and displays it to her in a way that she can understand.

Data Formats and Walled Gardens

“Tell me about yourself.”



But suppose they did not have pre-arranged consensus.

Data Formats and Walled Gardens

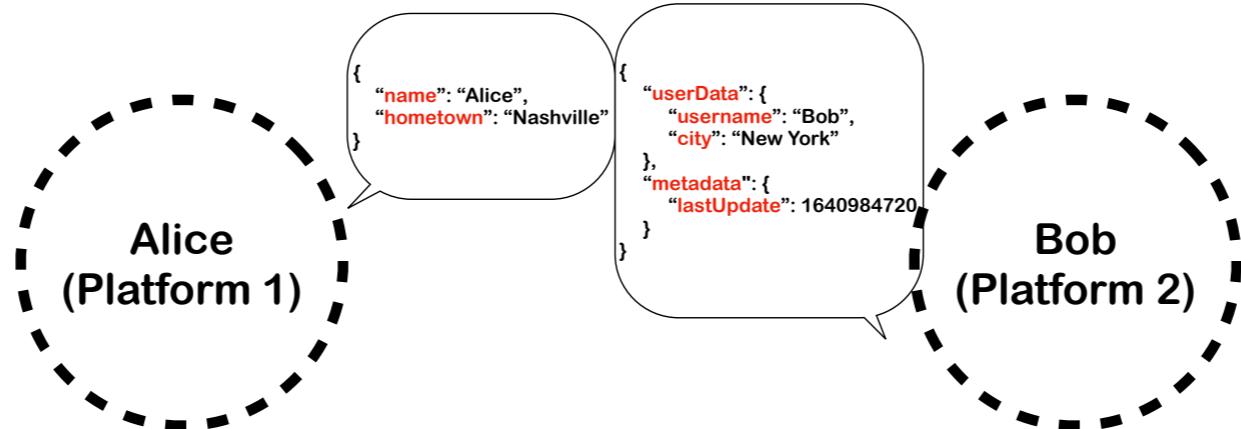
“Tell me about yourself.”



Bob uses one format,

Data Formats and Walled Gardens

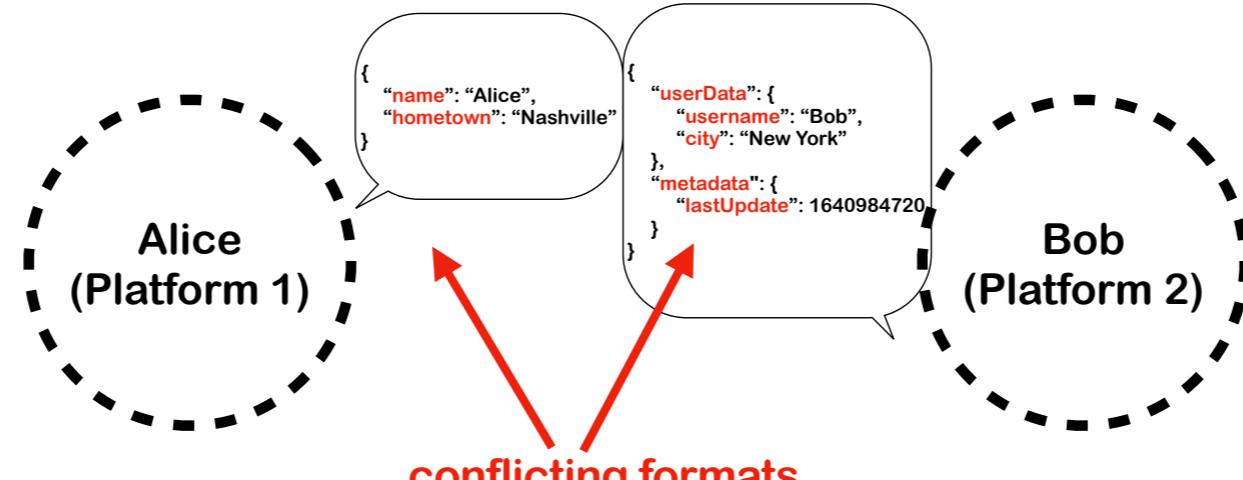
“Tell me about yourself.”



and Alice expects a different format.

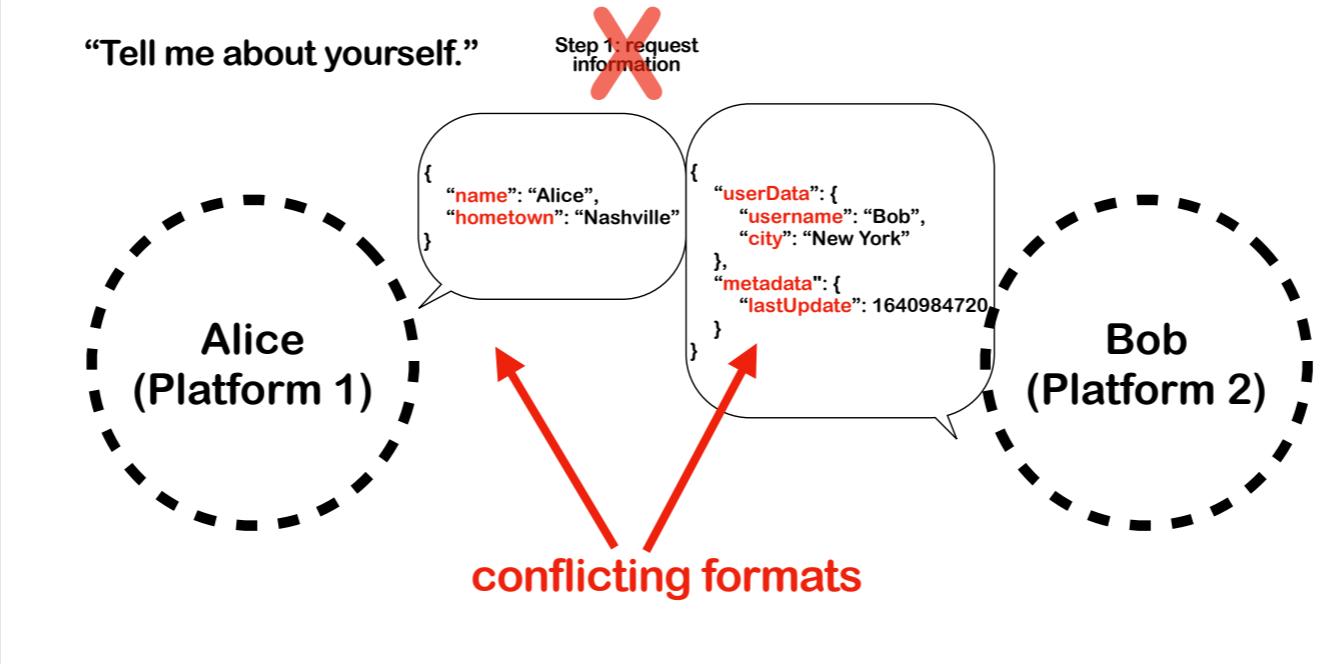
Data Formats and Walled Gardens

“Tell me about yourself.”



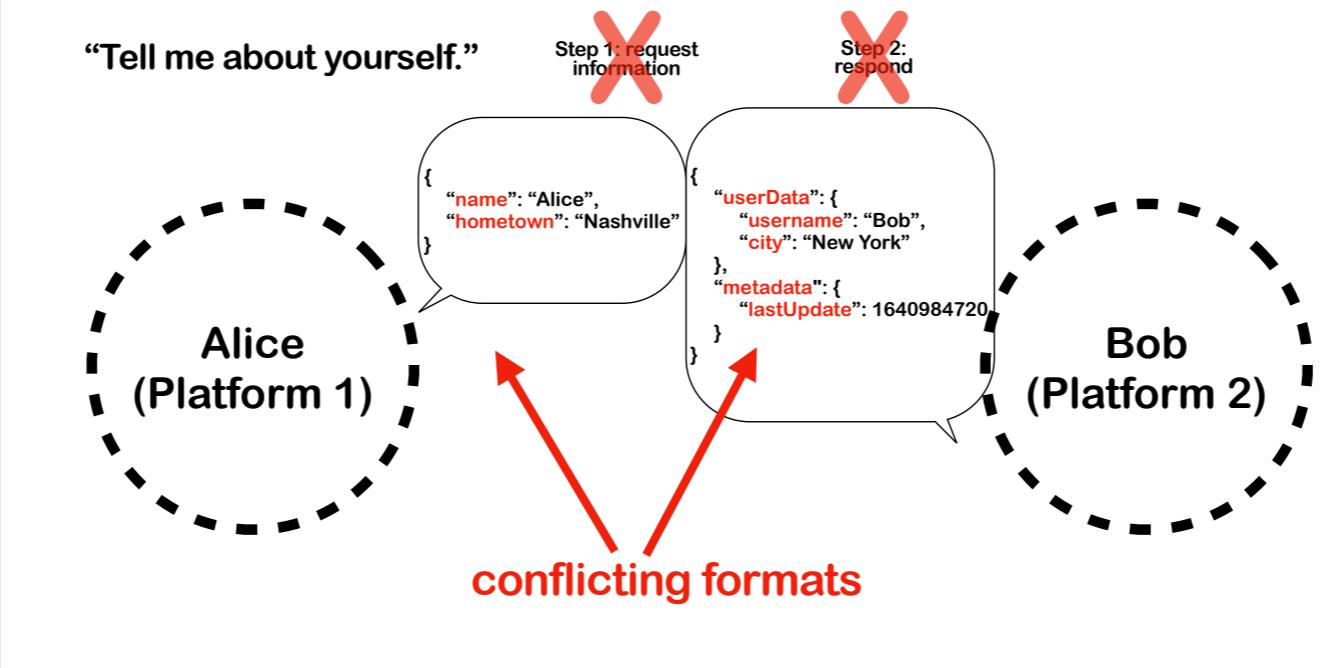
Their formats conflict, so her software can't interpret the file that he sent.

Data Formats and Walled Gardens



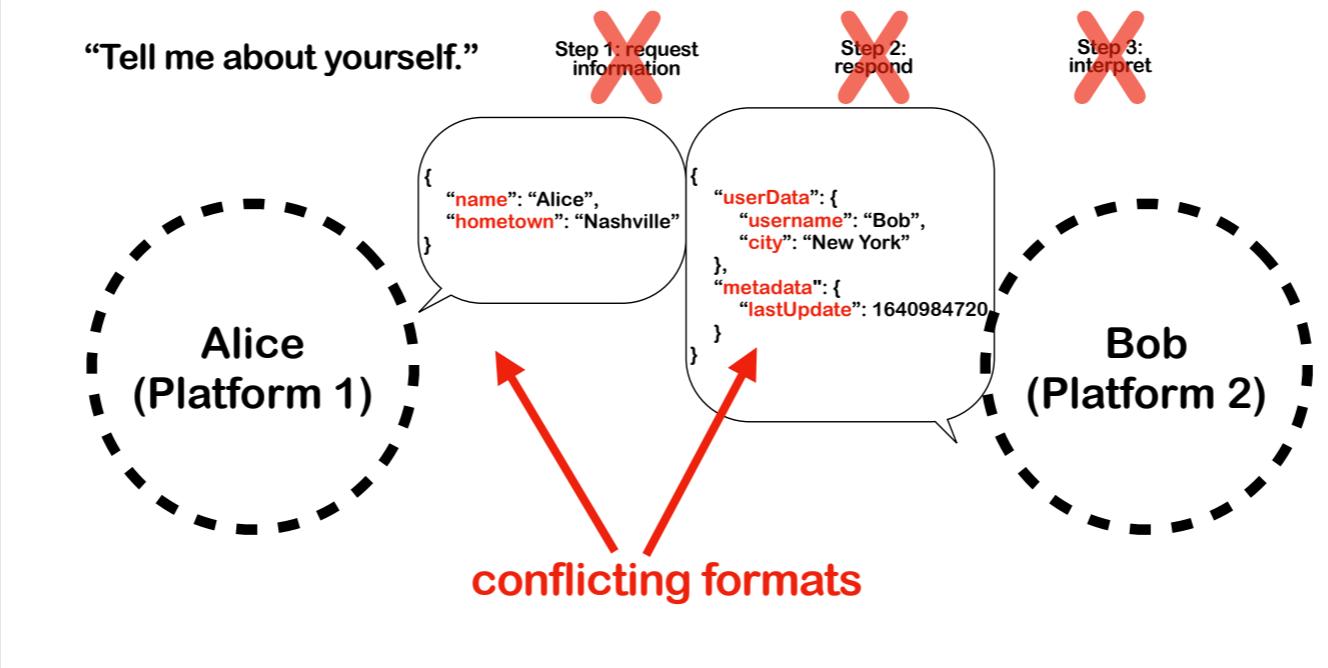
In fact, she never would have gotten past step 1, bc her software wouldn't have known what file or files to ask for.

Data Formats and Walled Gardens



If she had sent a request, his software probably would not have known what to do with it. So he couldn't do step 2.

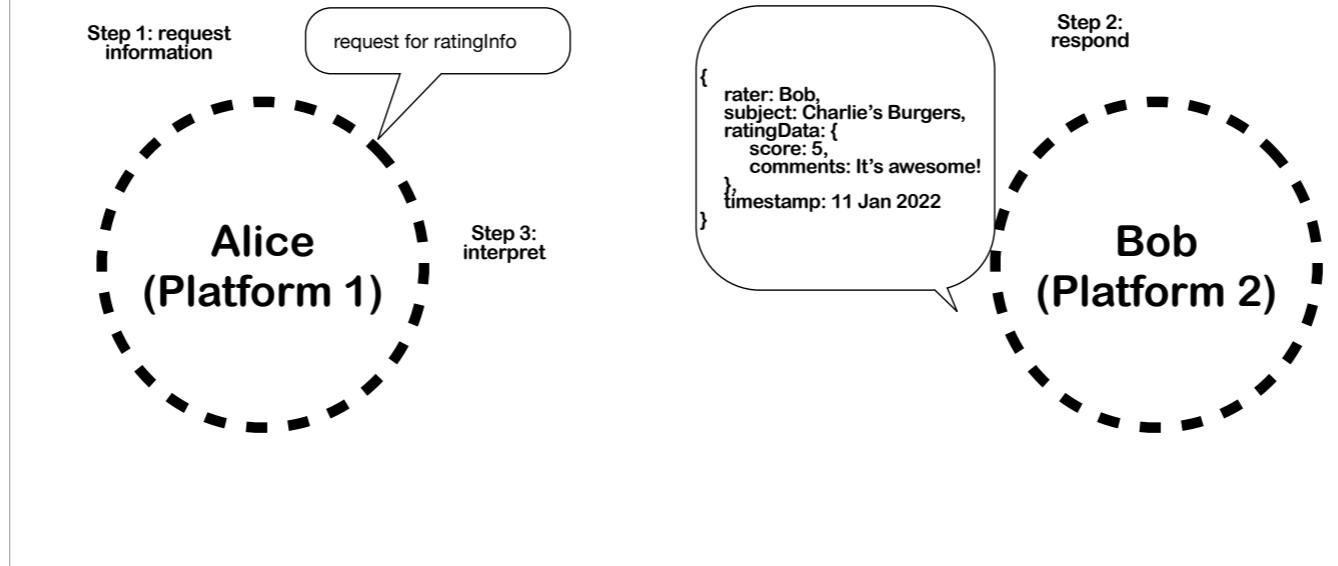
Data Formats and Walled Gardens



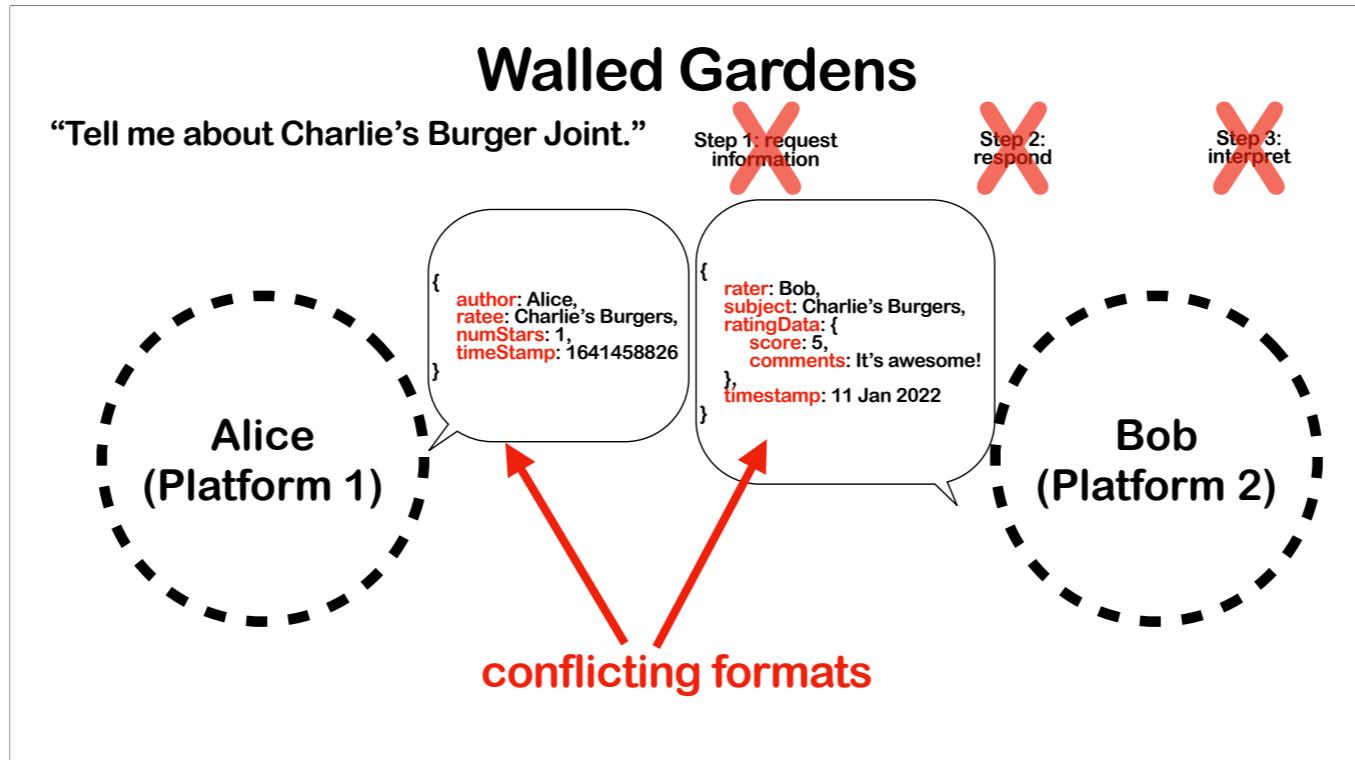
And as we have seen, step 3 is difficult at best. So none of the 3 steps are possible without a pre-arranged mutual understanding.

Walled Gardens

“Tell me about Charlie’s Burger Joint.”



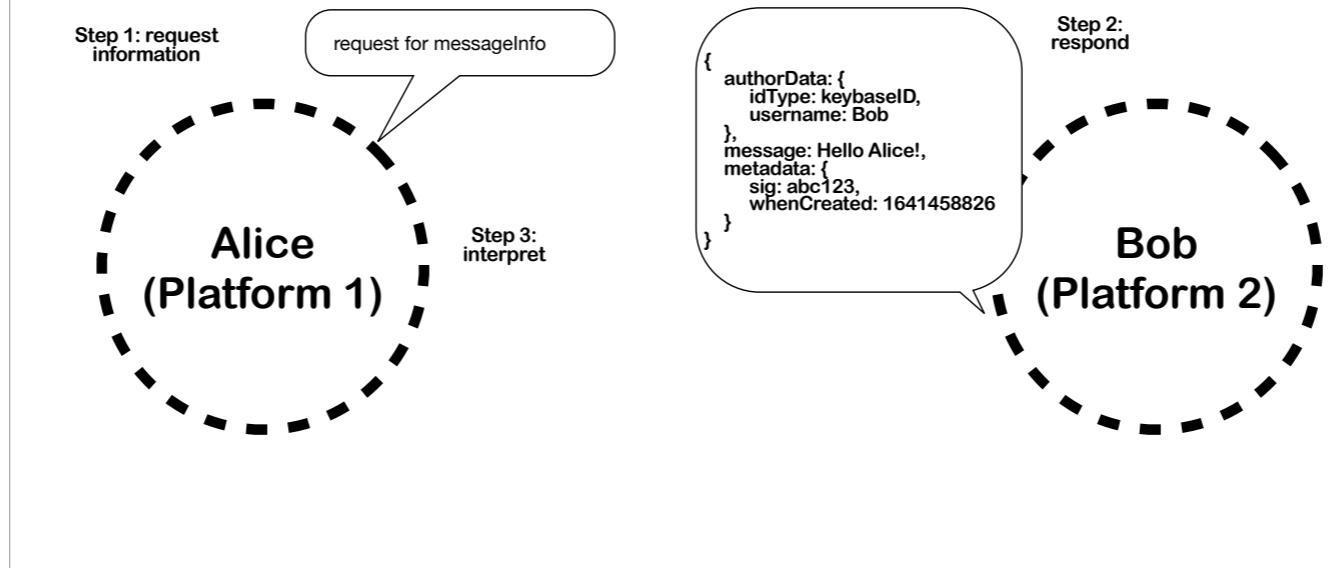
Another example: If she wants to ask what he thinks about a burger joint, we have the same scenario. She wants, essentially, to know his rating of the burger joint,



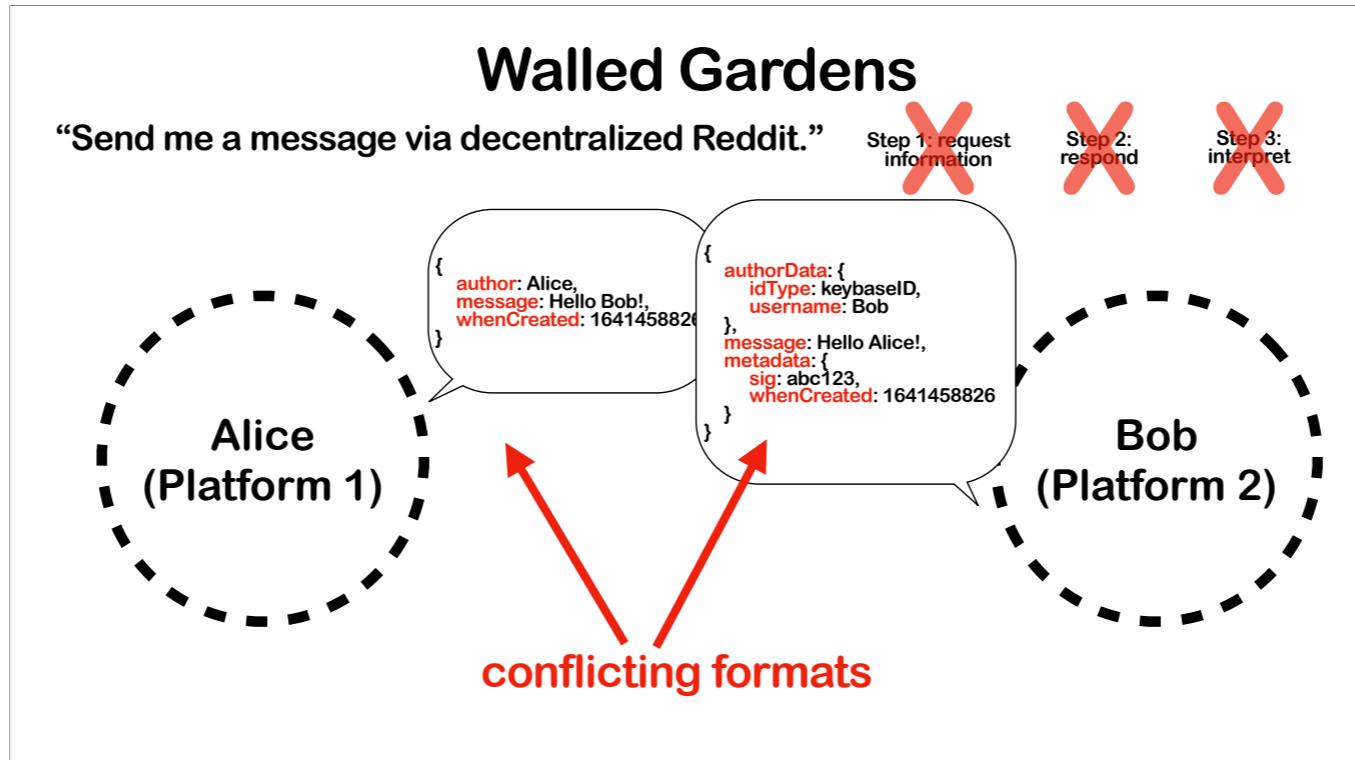
but she has no way to communicate to him the nature of the information she is looking for. Step 1 doesn't happen. Step 2 doesn't happen. Step 3 at best is difficult but it probably just doesn't happen.

Walled Gardens

“Send me a message via decentralized Reddit.”



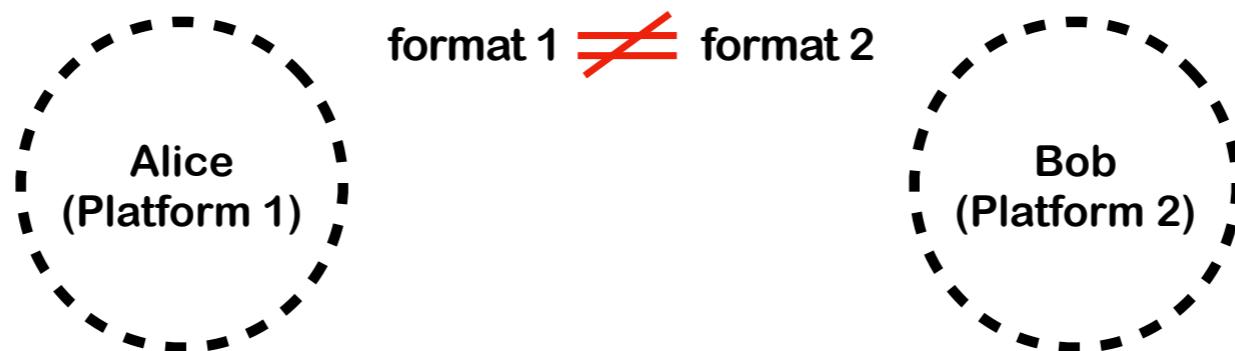
I could go on with countless examples. On whatever topic they want to communicate, like simply sending messages via a decentralized Reddit,



they would need to have pre-established consensus on all relevant data formats before they can do steps 1, 2, or 3.

Walled Gardens

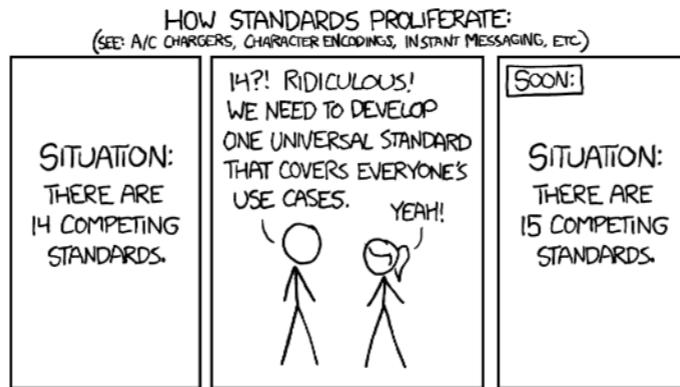
The essential problem leading to walled gardens is lack of a prearranged mutual understanding (consensus) on data format.



without a pre-established consensus on data formats, they continue to exist inside walled gardens.

(just in case you think the problem is solved by “universal standards” ...)

XKCD



Just in case you think the problem is solved by universal standards ... I bet many of you will have seen this cartoon by XKCD. Situation: there are 14 competing standards. Someone says that's ridiculous, we need to have just one Universal standard that everyone uses! But all that line of thinking manages to accomplish is to create one additional competing standard. With loose consensus, we won't have to worry about competing standards because our grapevines will generate consensus on whatever standards are important in whatever context that we need them.

[end 10]

11

The Principle of Loki

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.

[start 11]

As I have stated before, my proposed solution starts with the Principle of Loki: there must always be a method to connect data with its format that is simple, explicit, formalized, and hopefully widely used. The more data we subsume into this principle, the more success we achieve tearing down the walls. Convincing lots of people to implement this one simple idea — not necessarily the exact same software or implemented in exactly the same way; but simply to adopt the basic idea — is easier, and more doable, than trying to convince everybody to use the same sets of universal standards all the way down to their last details.

The Principle of Loki

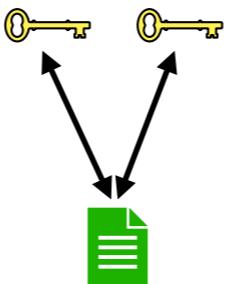
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



A piece of data, I represent in green; its format, in yellow.

The Principle of Loki

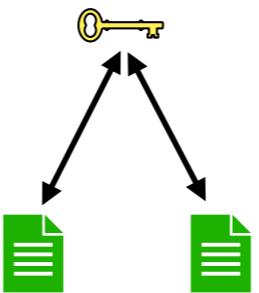
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The information on data formatting can be broken up into pieces,

The Principle of Loki

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



and conversely, one record of data format can, and typically will, be applicable to multiple files.

The Principle of Loki

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



So the principle, which is our starting point, can be stated quite succinctly. But how do we implement it? There may be multiple ways; but I think the implementation I present in this video is a good place to start.

The Concept Graph

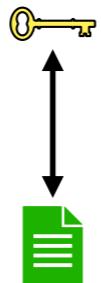
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The best way to implement this principle, at least in my assessment, is as follows:

The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



IMPLEMENTATION

- graph database

We organize data using a graph database,

The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



IMPLEMENTATION

- graph database
- nodes are files

where files are represented by nodes in the graph,

The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



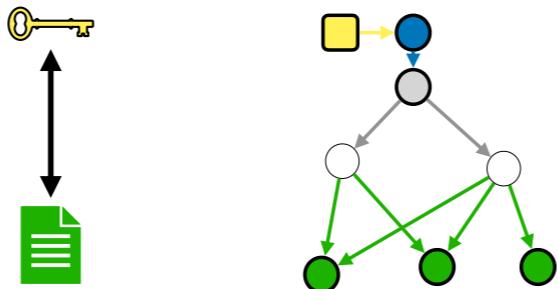
IMPLEMENTATION

- graph database
- nodes are files
- edges are specialized relationships between files

and edges represent specialized relationships between the files.

The Concept Graph

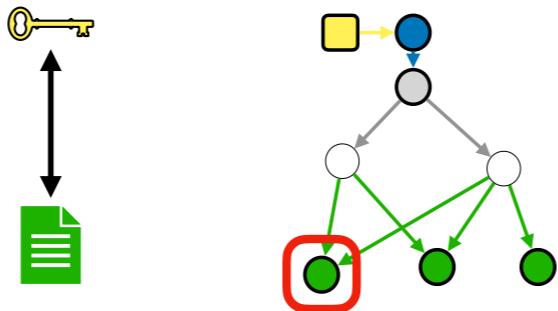
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



We then create a handful of basic relationshipTypes, denoted using differently colored arrows, and use them to connect nodes in such a manner, so that for any given file,

The Concept Graph

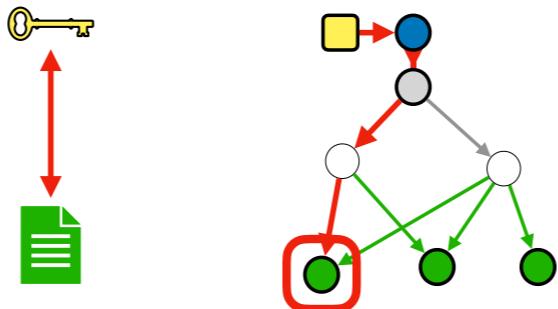
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



like the one highlighted in red, the formatting information for that file can be found consistently, accurately by tracing a certain well defined path, which I will call a Loki Pathway.

The Concept Graph

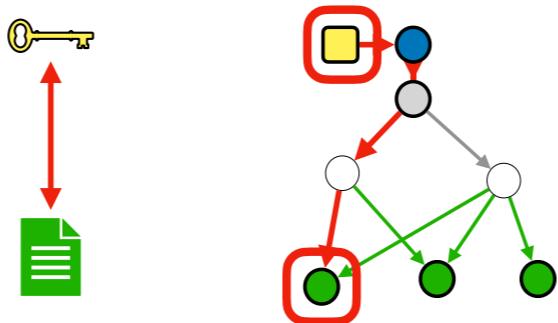
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



like this one highlighted in red,

The Concept Graph

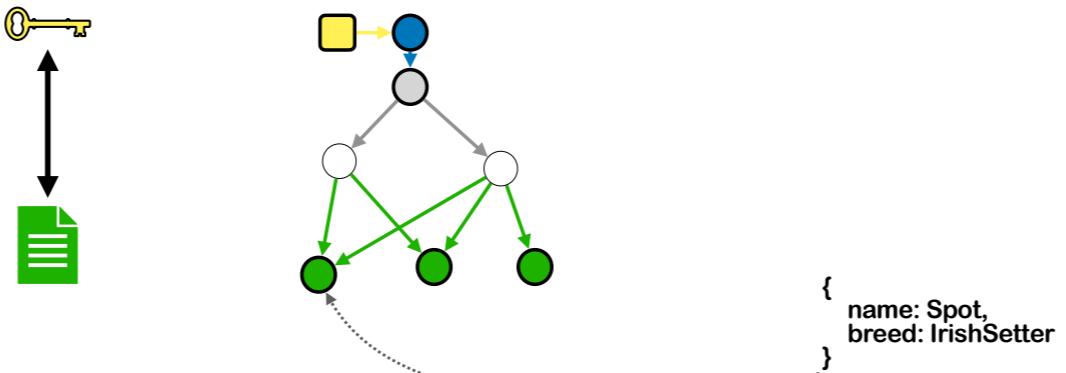
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



which connects it to the yellow node with formatting information that applies to the green nodes at the opposing end of the Pathway.

The Concept Graph

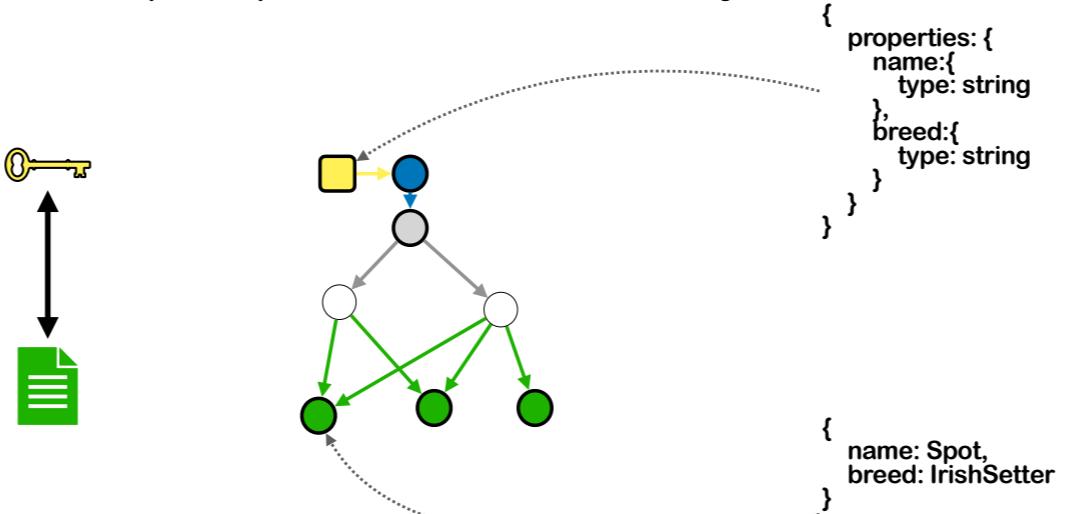
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



In this example, here's what one of the green nodes' files looks like; it has two properties, the first has the key: name, the value of which is a string, and the second property has the key breed, the value of which is also a string.

The Concept Graph

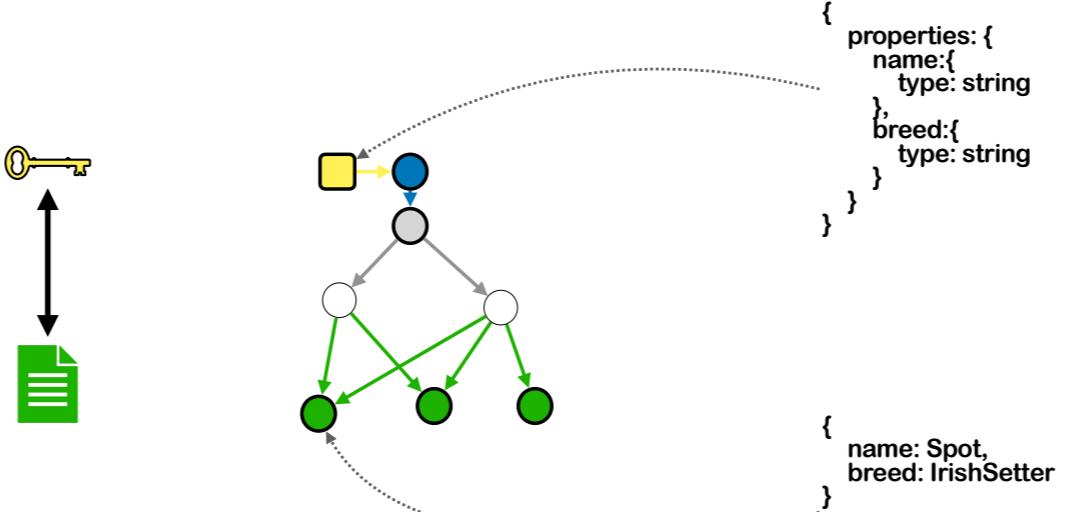
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



and here is the yellow node's file containing formatting information, basically what I just said: two properties, name and breed, each a string. This yellow box is the JSON Schema. Each of the files represented by the 3 green nodes are expected to follow the format spelled out by the yellow node. If it doesn't, then there's an error somewhere, that must be addressed.

The Concept Graph

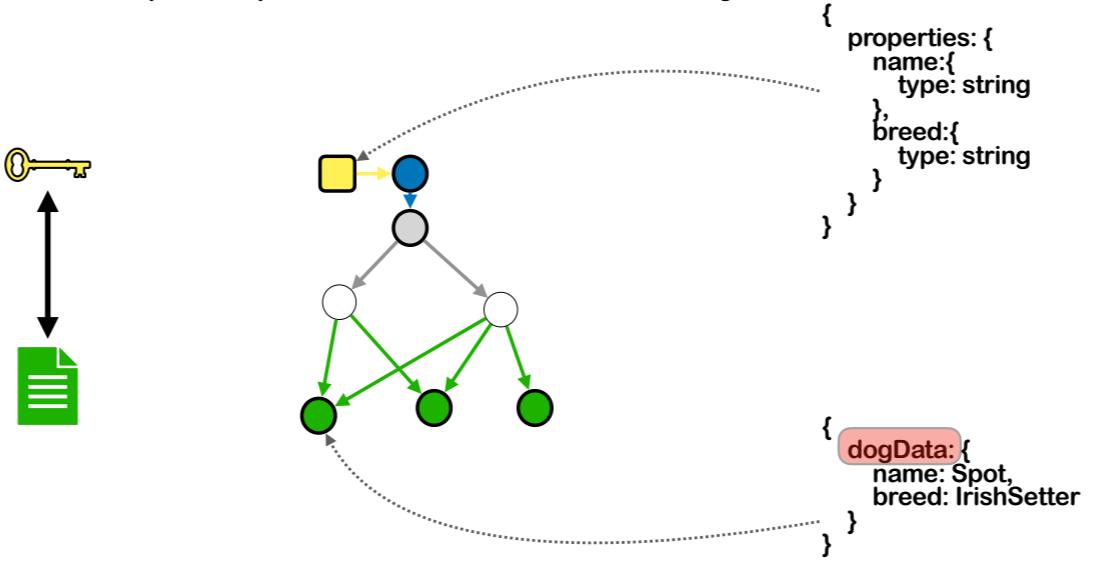
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



Another way to phrase things, would be to say that the yellow node describes a pattern; if a file matches that pattern — meaning: if the file is formatted according to the rules in the yellow node JSON Schema — then that file belongs here, in this concept, displayed as a green node. You could even imagine that if the software comes into contact with a new file, and needs to figure out where to put it, it runs through all of the yellow node JSON Schemas in its database, looking to see whether the file's format makes it a match for any of these patterns. Each Yellow box can be thought of as the description of a Pattern. Remember this later in this video when I discuss the Pattern Recognition Theory of Brain.

The Concept Graph

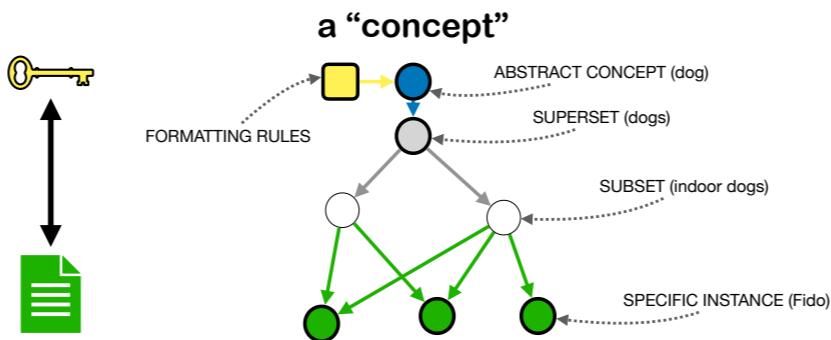
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



As a side note, I often like to package the properties for any given file under a heading property, such as, in this case, dogData, since we are working with the concept for dog. I haven't shown it here, but the JSON Schema would be modified slightly to accommodate this change.

The Concept Graph

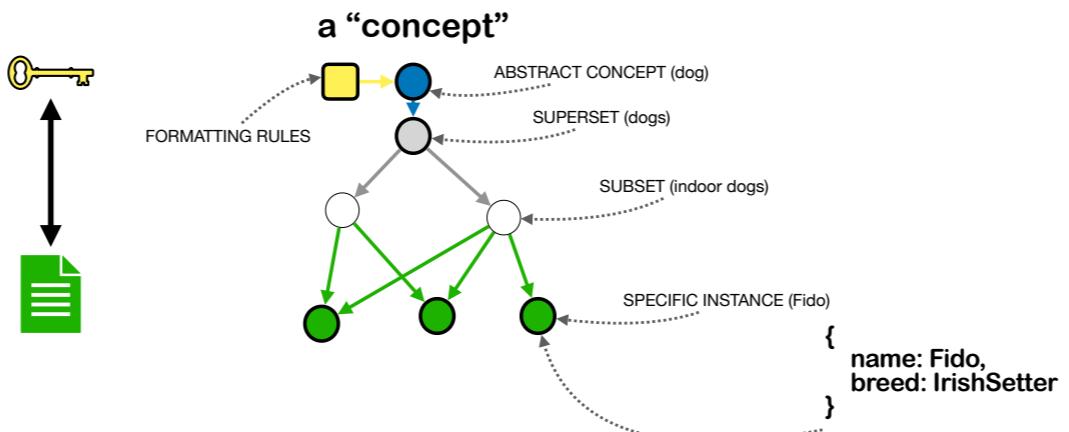
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The entire thing as a unit, I call a concept: loosely defined, a concept is composed of a single JSON Schema, in yellow, plus every Loki Pathway emanating from that JSON Schema, and every node along those pathways. There are several types of nodes, indicated by colors and shapes, depending on where they are along the Loki Pathway. A concept can represent anything — dog, user, rating — but let's take this one for dog and traverse the Loki Pathway from bottom to top.

The Concept Graph

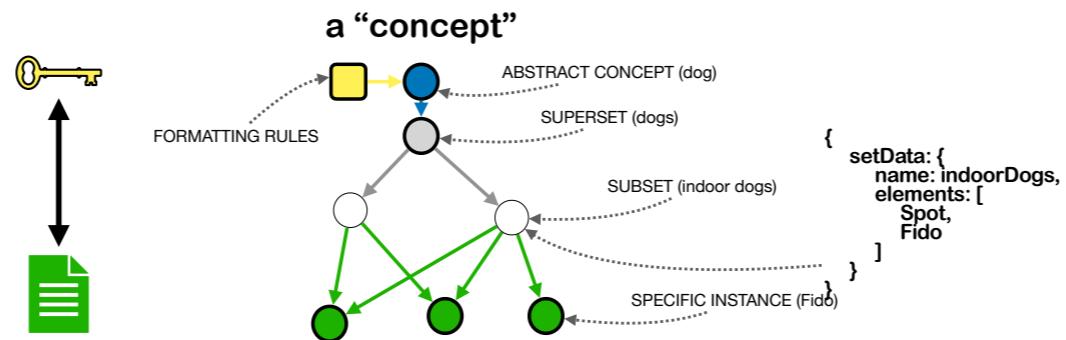
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



At the bottom we have the green nodes: each represents an individual dog: Fido, Spot, etc.

The Concept Graph

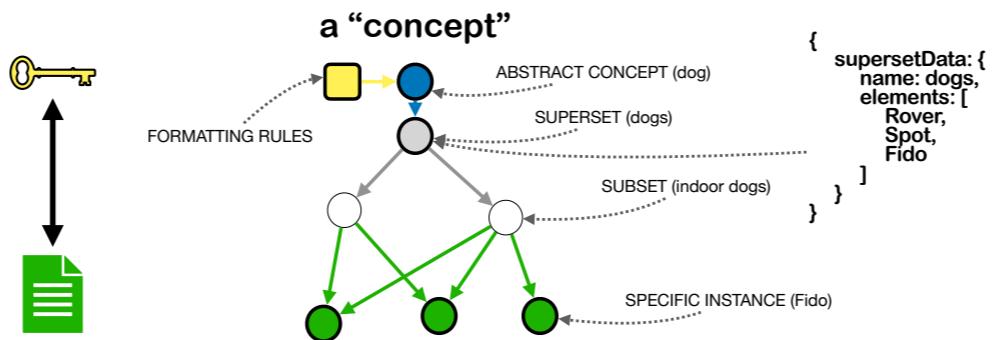
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



Moving upwards: The white nodes represent sets of dogs, for example: “indoor dogs” or “poodles” or “female dogs” or “rescue dogs” or however the user wants to group them.

The Concept Graph

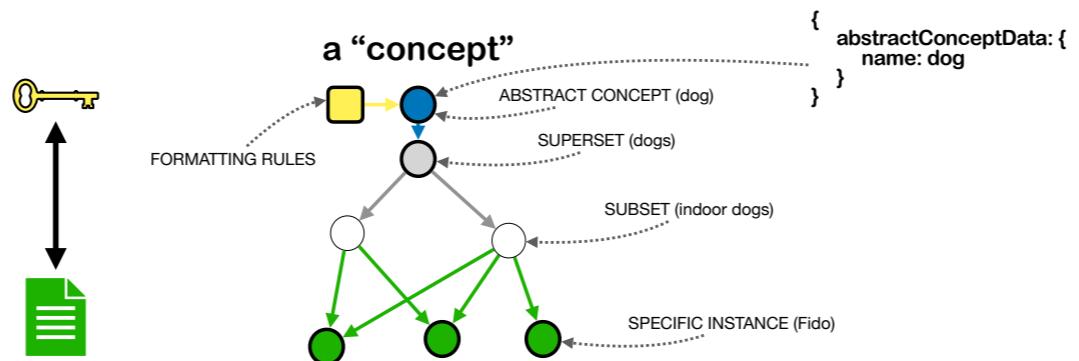
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The grey node is the superset of all dogs in this database.

The Concept Graph

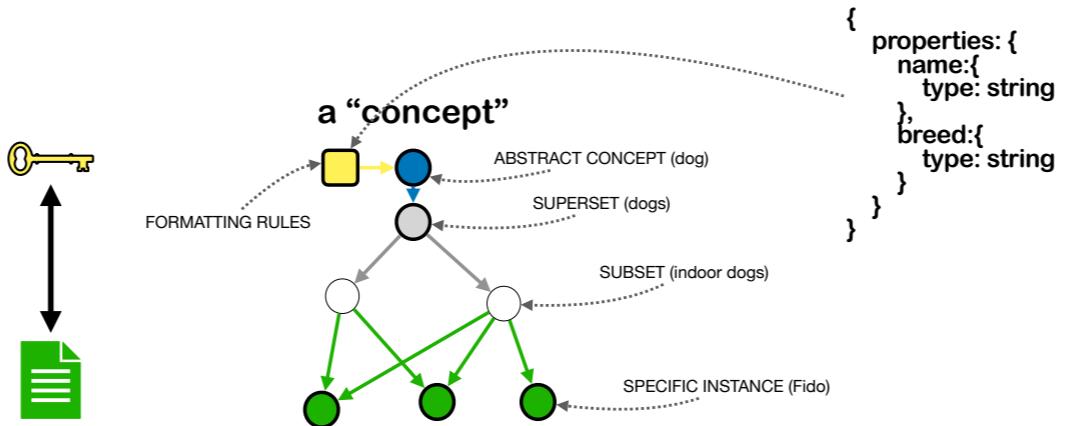
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The blue node represents the abstract concept of a dog.

The Concept Graph

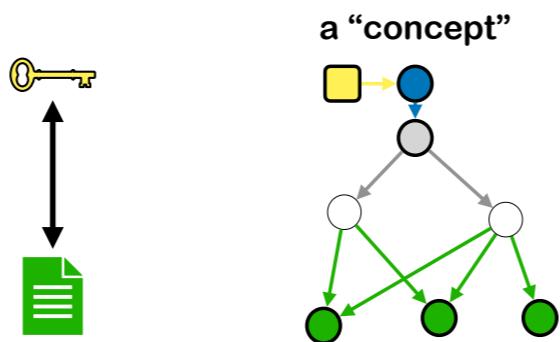
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



And finally the yellow node is the one with the JSON Schema formatting rules.

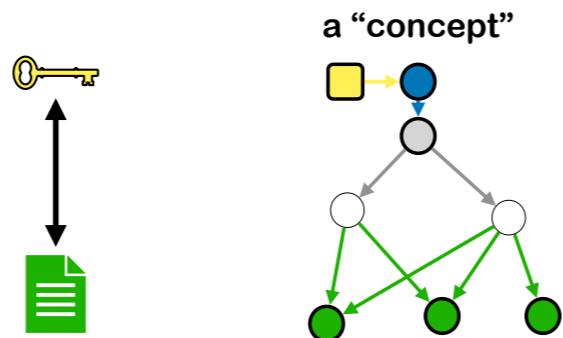
The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



The Concept Graph

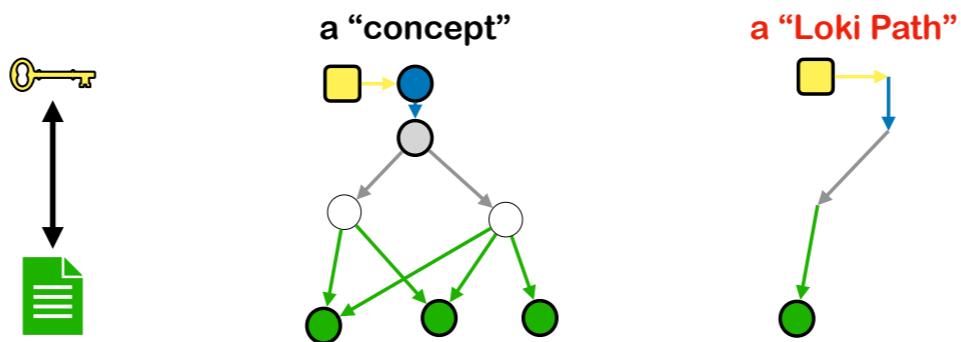
Principle of Loki: There must always be a **method** to connect data with its format that is: simple, explicit, formalized, and widely used.



Summarizing: the Principle of Loki calls for a method to connect a file with its formatting information. And in my implementation, the method to accomplish this is to follow

The Concept Graph

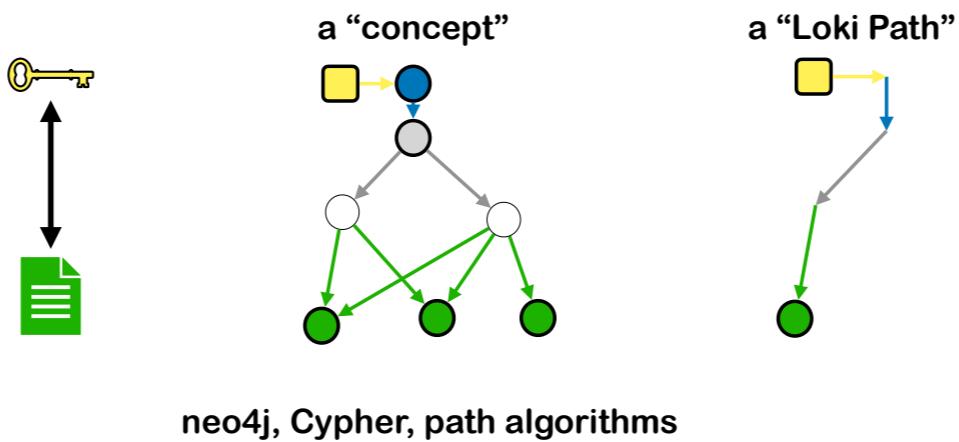
Principle of Loki: There must always be a **method** to connect data with its format that is: simple, explicit, formalized, and widely used.



a particular path through a graph. Which I call a Loki Pathway. It is simple, explicit, and well defined.

The Concept Graph

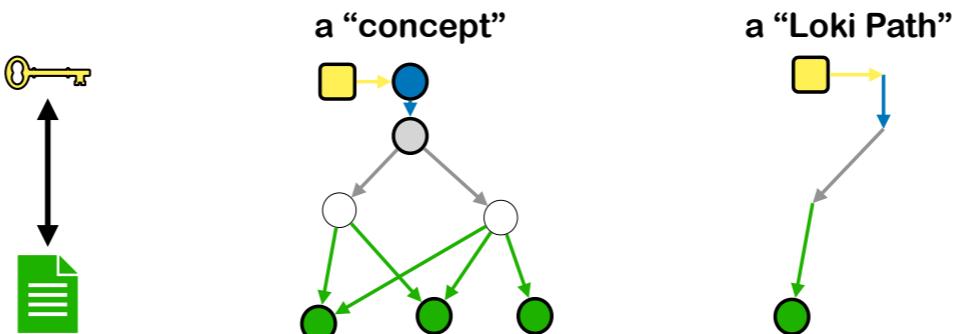
Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



I'd like to point out that there are several open source libraries for graph databases. The one I am most familiar with is neo4j. It comes with a query language, called cypher, which contains, among other things, path algorithms. So it is a simple matter to write a Cypher query to find every Loki Pathway within a graph database: just start at a yellow node, then look for one yellow arrow, one blue arrow, any number of grey arrows, and then ending in one green arrow.

The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



```
MATCH (a1:JSONSchema)-[:isTheJSONSchemaFor]-(a2:concept)-[:isTheSupersetFor]-(a3:superset)-[:subsetOf*0..5]-(a4:set)-[:isASpecificInstanceOf]->(a5)
WHERE [conditions]
RETURN a1,a5
```

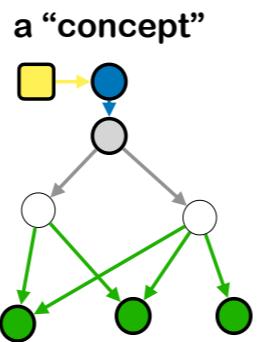
And just for curiosity sake, here is a Cypher Query that would be used to look for Loki Pathways, and find every pair of a data file with a matching formatting file.

[end 11]

12

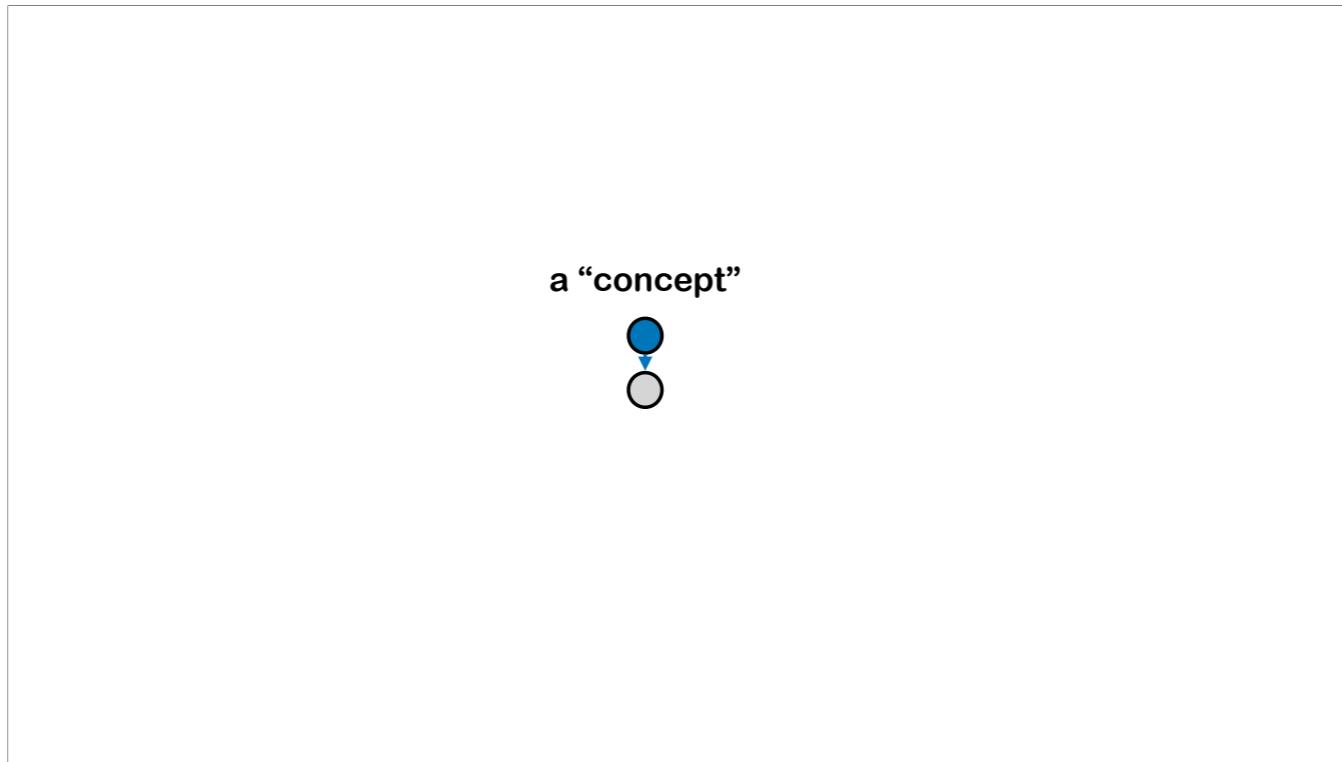
The Concept Graph

Principle of Loki: There must always be a method to connect data with its format that is: simple, explicit, formalized, and widely used.



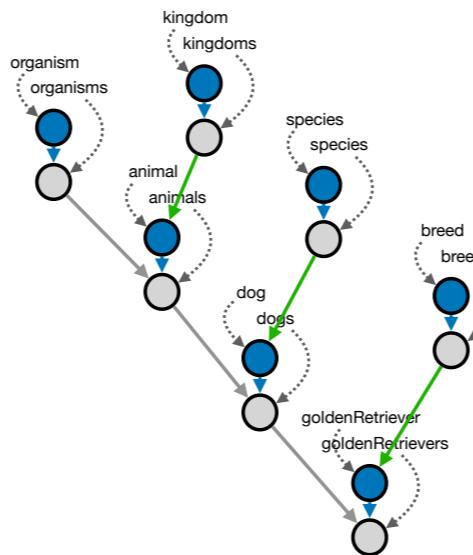
[start 12]

I've shown you the makeup of an individual concept. Why do I call it the concept graph?



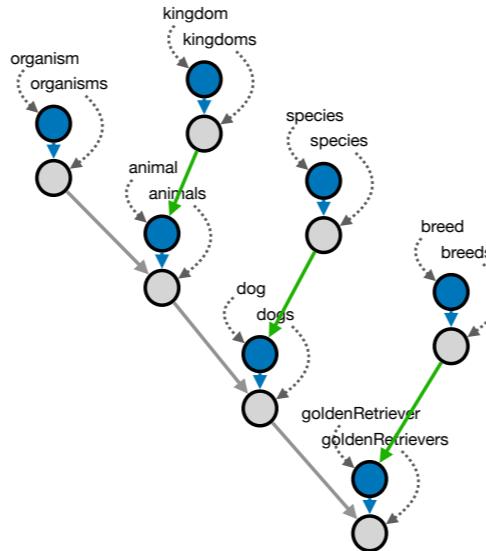
The core of any concept is made up of these two nodes, the abstract concept node in blue and the superset node in grey.

multiple interrelated concepts = a Concept Graph



If you take a bunch of related concepts and establish relationships between them, like this, then you have a Concept Graph. As you can imagine, it can grow to be pretty large, pretty quickly.

The Concept Graph App



The Concept Graph app is made to manage all of this stuff: the concepts, their interrelationships, their JSON files, the graph database, communication with other users, interface with the grapevine, everything.

The Concept Graph App

The Concept Graph App

- electron
- IPFS
- REACT.js
- neo4j
- JSONSchema
- sqlite3
- vis.js

These are some of the open source tools that I have incorporated into the Concept Graph app. They are all also being used for the Grapevine app.

The Concept Graph App

- electron
- IPFS
- REACT.js
- neo4j
- JSONSchema
- sqlite3
- vis.js
- **BTC/LN**

I do envision incorporating BTC/LN in the future, although I haven't tried coding any of that yet. I will show you later in this talk how and why micropayments will fit in to this system.

The Concept Graph App

Front end

So: the way the CG functions on the front end:

The Concept Graph App

Front end

- User can create and edit Concepts and organize into Concept Graphs

The user will be able to create individual concepts and organize them into Concept Graphs.

The Concept Graph App

Front end

- User can create and edit Concepts and organize into Concept Graphs
- all actions are INTUITIVE

all of the actions involved in this process should be quite intuitive

The Concept Graph App

Front end

- User can create and edit Concepts and organize into Concept Graphs
- all actions are INTUITIVE
- no need for developer-level skills

meaning that there should be NO NEED for the user to have developer-level skills.

The Concept Graph App

Front end

breaking this down into smaller pieces, the user will be able to:

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)

create concepts;

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)

add elements (meaning, examples for specific concepts; like Fido and Spot are examples of dog)

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)
- add fields (properties) (name, owner, etc)

add fields, or properties: like - each dog has a name, each dog has an owner, etc.

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)
- add fields (properties) (name, owner, etc)
- organize into subsets (Irish Setters, etc)

organize them into subsets, like - Irish Setters is a subset of dogs, etc.

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)
- add fields (properties) (name, owner, etc)
- organize into subsets (Irish Setters, etc)
- create relationships between concepts (dog — breed)

and then create relationships between one concept and another. Like: one property of a dog would be to specify its breed.

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)
- add fields (properties) (name, owner, etc)
- organize into subsets (Irish Setters, etc)
- create relationships between concepts (dog — breed)

CONCEPTUALIZATION: organizing a group of related concepts (e.g. all or part of an app's database) into a concept graph

The process of doing all of these steps, is something that I call CONCEPTUALIZATION. It means creating a group of concepts and then tying them all together with various basic relationships. And a handful of basic relationship types goes a long way.

The Concept Graph App

Front end

- create concepts (dog, cat, animal, breed, etc)
- add elements (Fido, Spot, etc)
- add fields (properties) (name, owner, etc)
- organize into subsets (Irish Setters, etc)
- create relationships between concepts (dog — breed)

CONCEPTUALIZATION: organizing a group of related concepts (e.g. all or part of an app's database) into a concept graph

If you are doing this to organize an app's database, you can do this with the entire database, or you can do a partial conceptualization, dealing with just a subset of the app. I'll discuss later about why you might do that.

[end 12]

13

Conceptualization: Creation of a Concept Graph

[start 13]

I'm going to walk through the process of conceptualizing something simple — dog, animal, etc — to give you an overview of what conceptualization entails.

Conceptualization: Creation of a Concept Graph

make concepts:

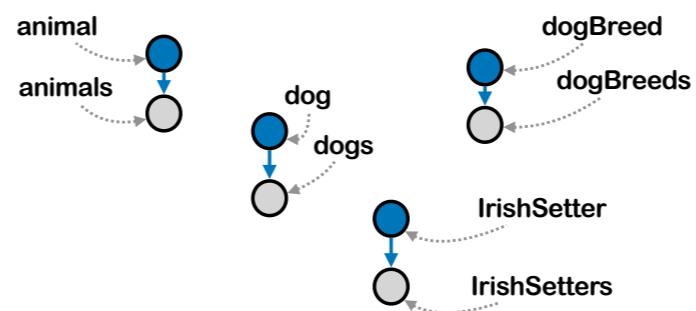
- animal
- dog
- dogBreed
- IrishSetter

The first step is to make some concepts. All you need is a name to start with.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

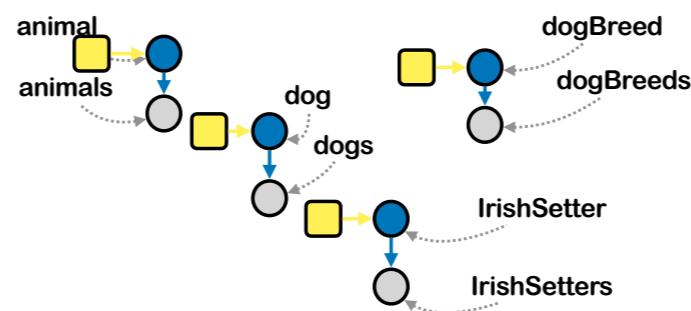


On the back end, the CG creates the corresponding files, nodes and relationships.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

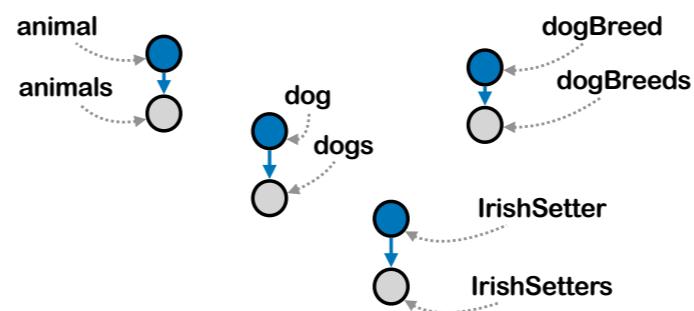


I won't always show it, but the corresponding JSON Schema files also get created at this step.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter



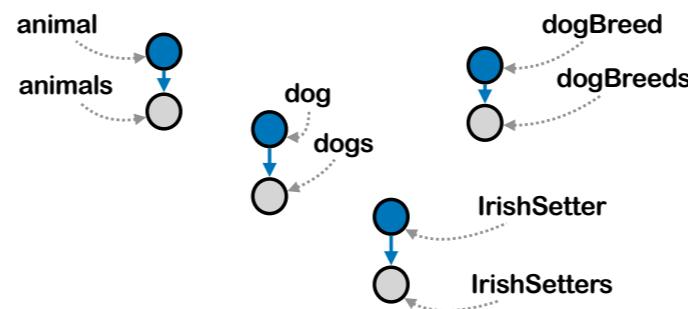
Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido



Next, the user adds examples of individual concepts,

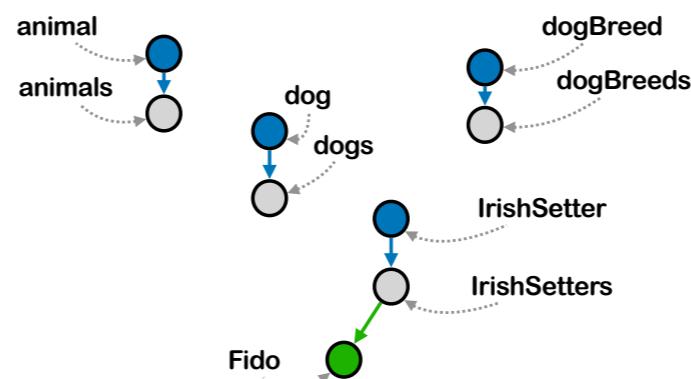
Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido



such as: Fido as an example of an Irish Setter,

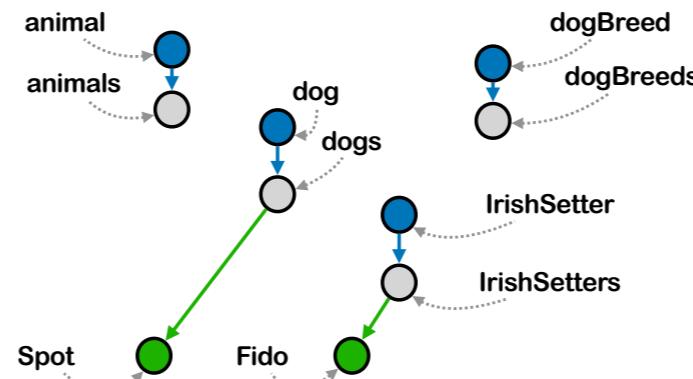
Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido



and Spot as an example of dog. Assuming here that the user doesn't know whether Spot is an Irish Setter or some other breed.

Each node also has a starter JSON file at this point.

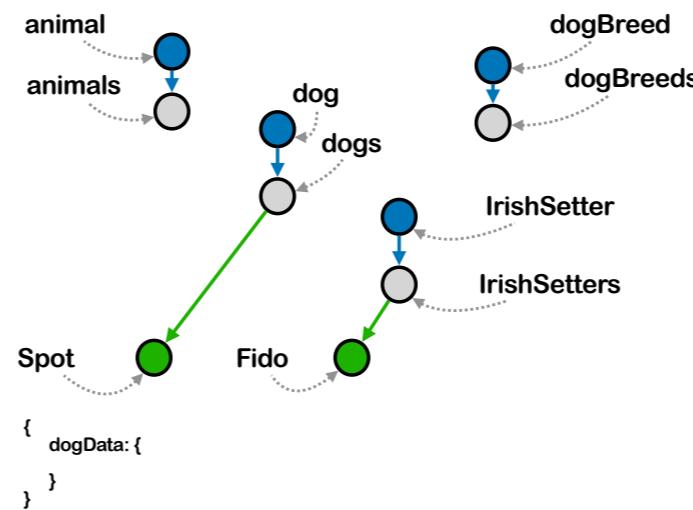
Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido



Spot's JSON file would look like this. Not much in it yet, as you can see.

Conceptualization: Creation of a Concept Graph

make concepts:

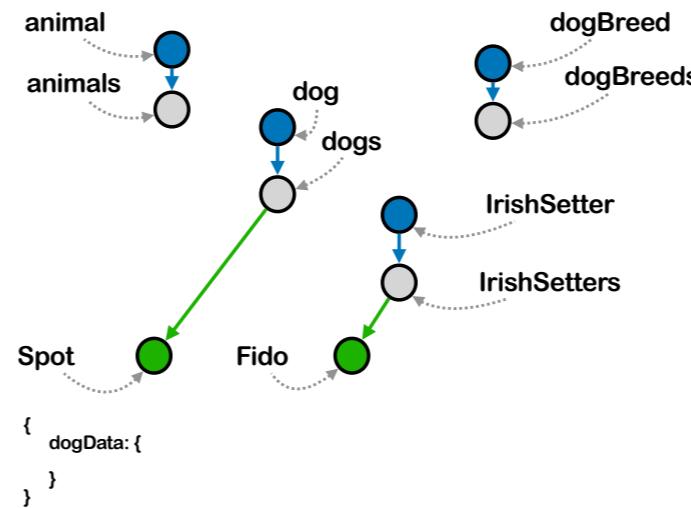
- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed



Next, the user adds fields for individual concepts. For example: name, owner, and breed are fields for dogs.

Conceptualization: Creation of a Concept Graph

make concepts:

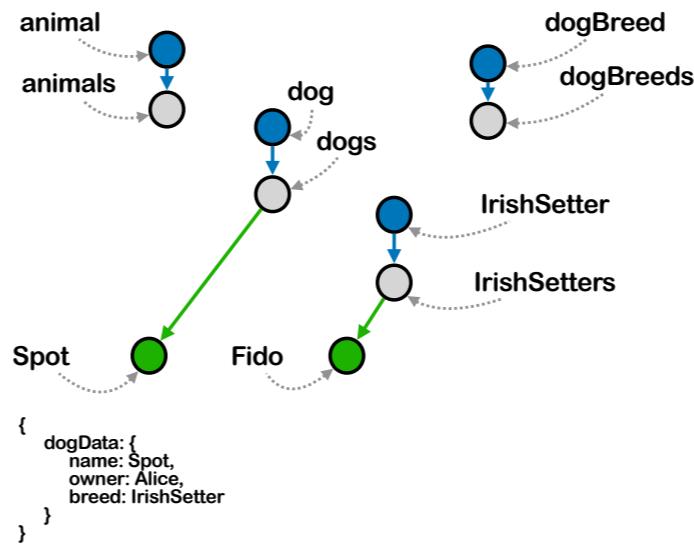
- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed



This allows the user to fill out those fields for each example, as you see here for Spot: name is Spot, owner is Alice, and breed is Irish Setter.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

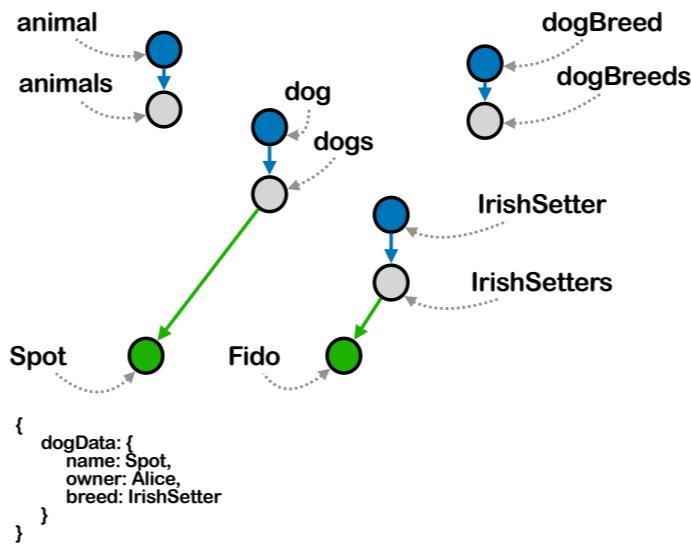
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



Subsets can also be added.

For example, cats and dogs are subsets of animals.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

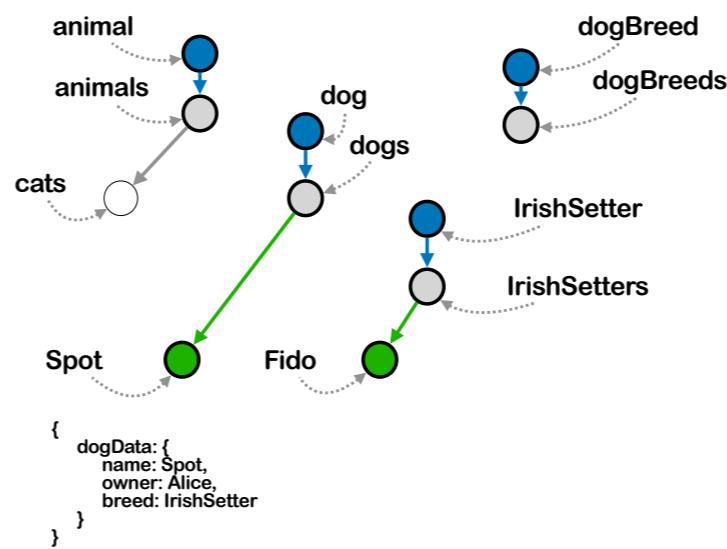
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



A new node must be created for cats,

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

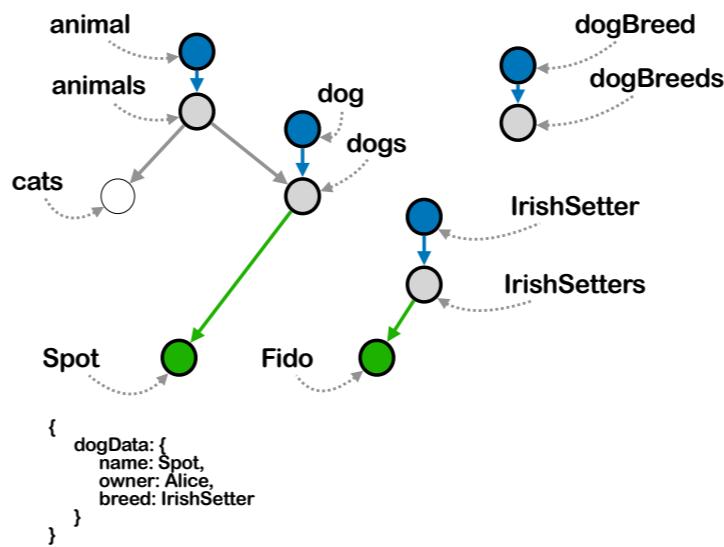
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



but for dogs, there is already a node for the superset of all dogs as part of the concept for dogs. The two grey arrows connecting animals to the node for cats and the node for dogs indicate a subset relationship.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

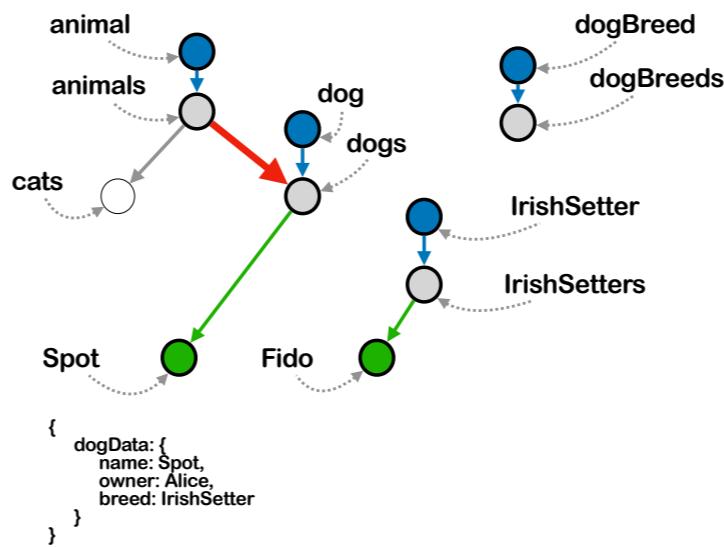
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



The act of making dogs a subset of animals means that we have created a new Loki pathway, which places Spot as being

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

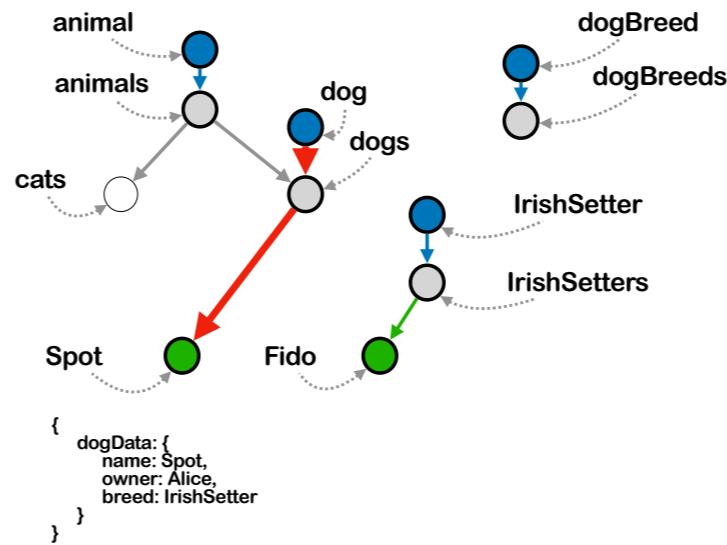
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



not only an example of the dog concept,

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

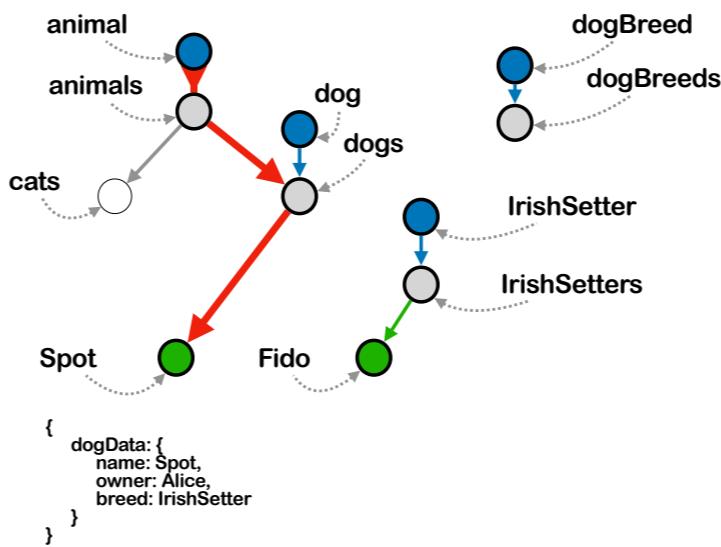
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



but also an example of the animal concept.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

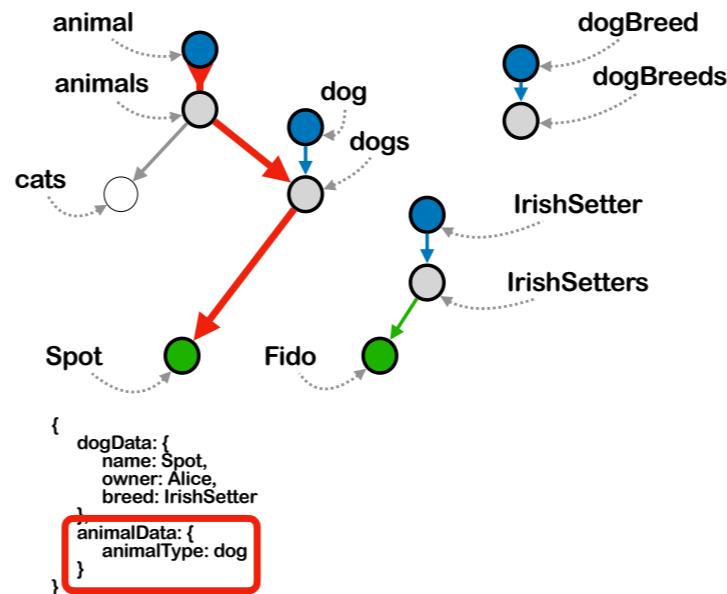
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



As a result of this new Loki Pathway, the JSON file for Spot is modified to contain an additional property, one corresponding to the concept for animals.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

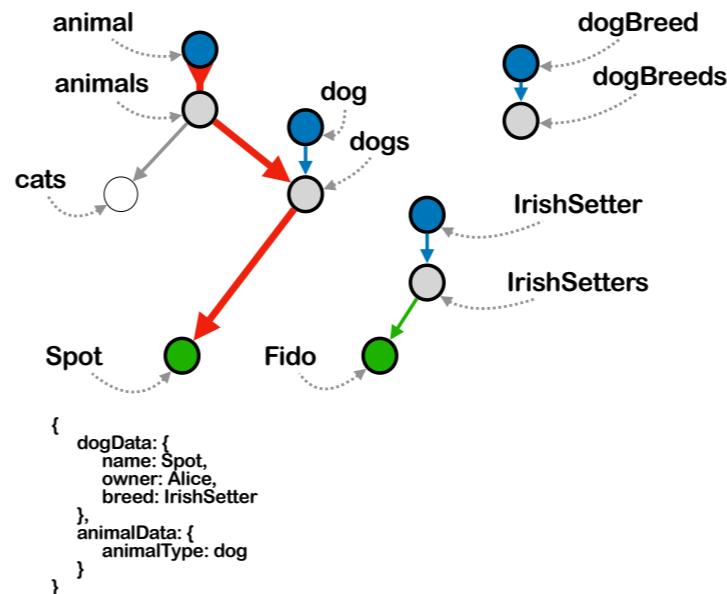
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



As you can see, the Loki Pathway really is the primary organizing principle for the Concept Graph.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

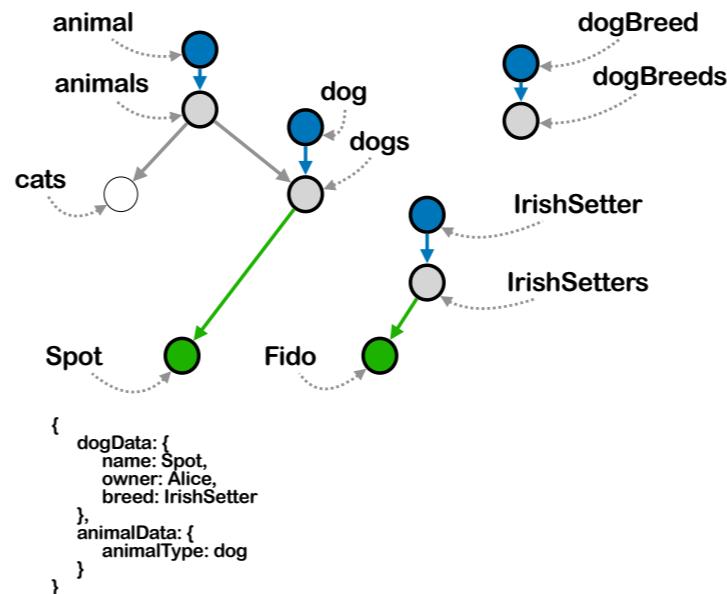
- Spot
- Fido

flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals



Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

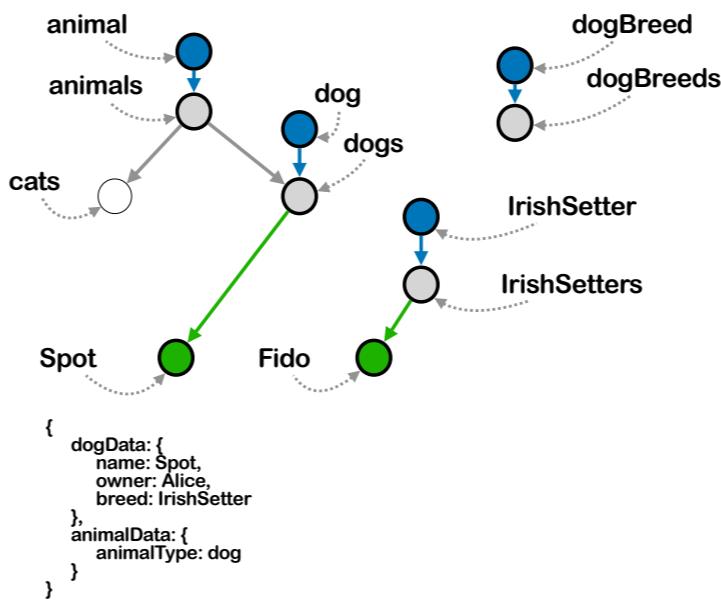
flesh out each concept with fields:

- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:



Finally, the user will add relationships between concepts.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

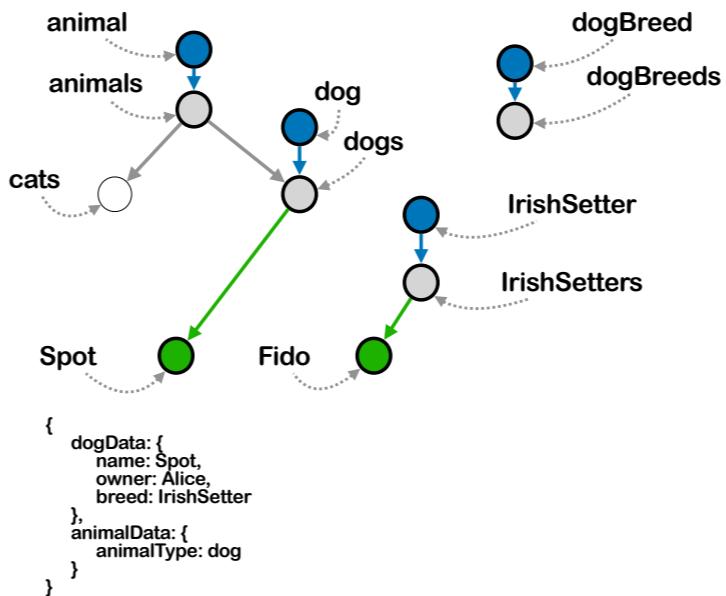
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed



IrishSetter is an example of dogBreed

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

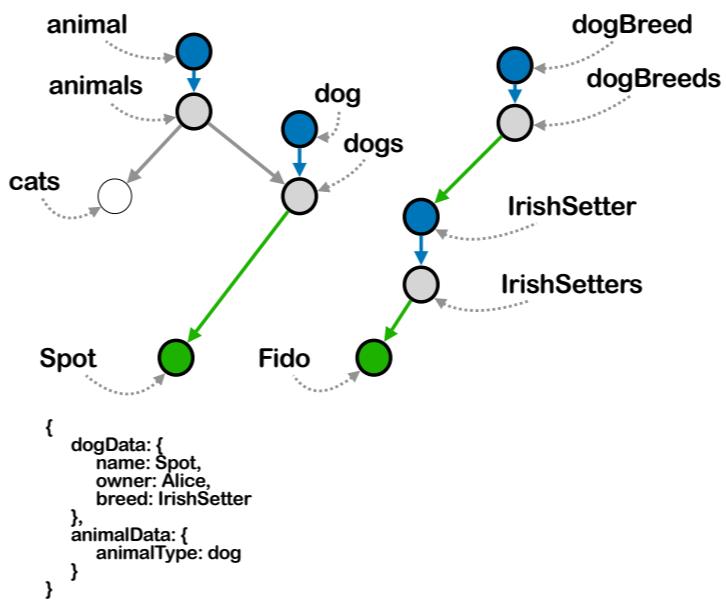
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed



Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

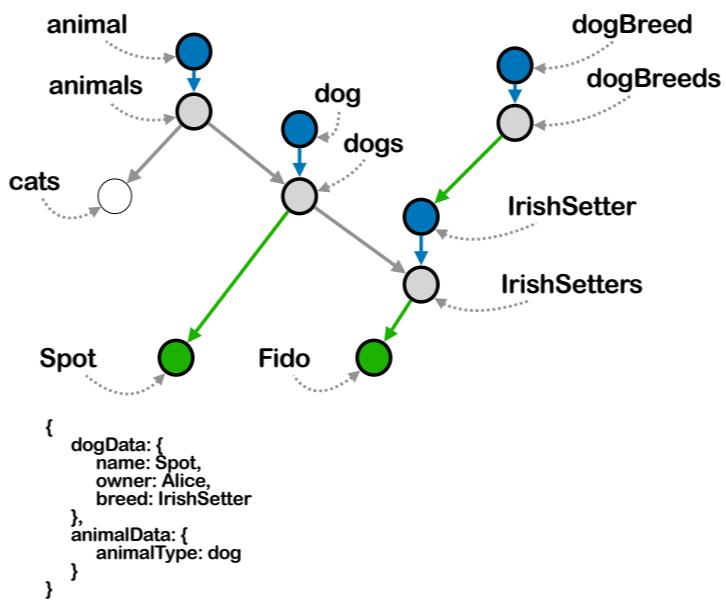
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed
- IrishSetters is a subset of dogs



and IrishSetters as a subset of dogs.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

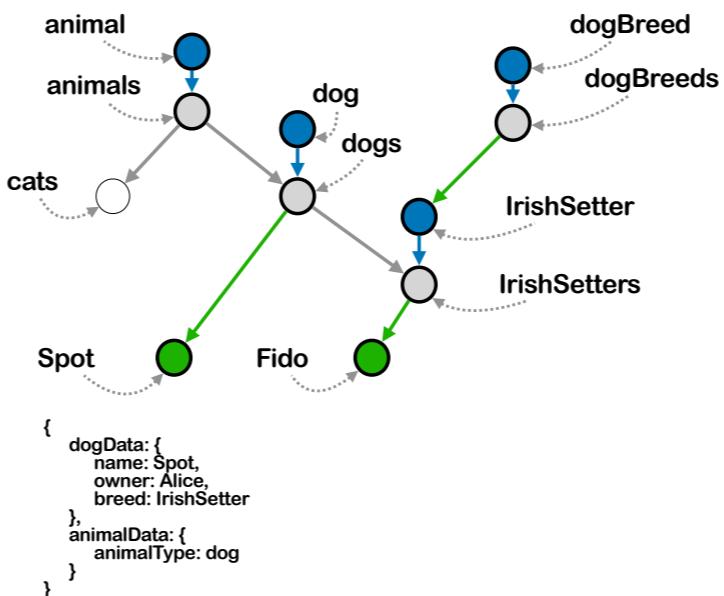
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed
- IrishSetters is a subset of dogs
- dogs is a subset of animals



Dogs as a subset of animals was added at an earlier step.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

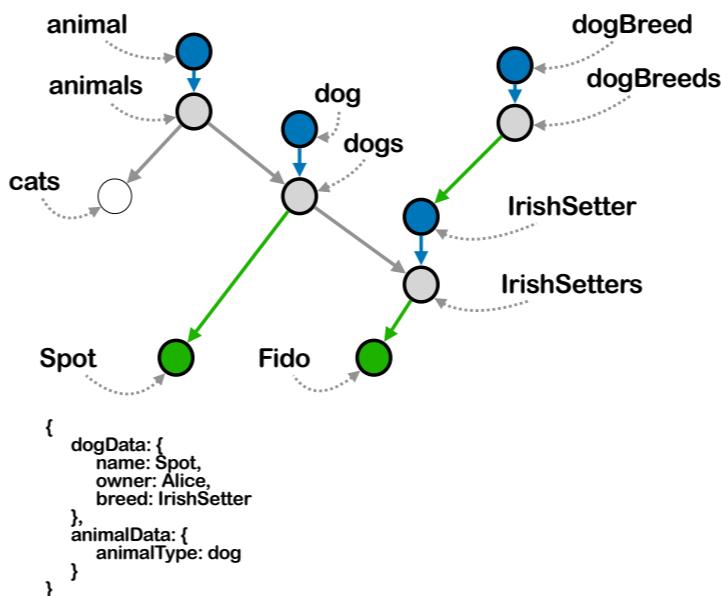
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed
- IrishSetters is a subset of dogs
- dogs is a subset of animals



As you can see, the user has created a concept graph through a series of steps that are all very intuitive. No need for the user to be a programmer or to know anything about JSON files, or even to know what J-S-O-N stands for.

The back-end maintains the graph database, generates JSON files, searches for Loki Pathways. You can drive a car without knowing what a carburetor is, and here the user can operate the CG without necessarily knowing what's happening underneath the hood. And that's good bc there's a lot going on under the hood. Take a look at dogs vs cats. Dogs was introduced as a concept, then related to animals. Cats was added as a subset. The user might decide later on to expand the cats subset into a full fledged concept, just like dogs. In this way, concepts can grow incrementally. Which is exactly how knowledge is gained in the real world: one small piece at a time. There's no reason this has to be difficult for the user, at the front end. Every incremental step here is intuitive at the front end.

Conceptualization: Creation of a Concept Graph

make concepts:

- animal
- dog
- dogBreed
- IrishSetter

flesh out each concept with examples:

- Spot
- Fido

flesh out each concept with fields:

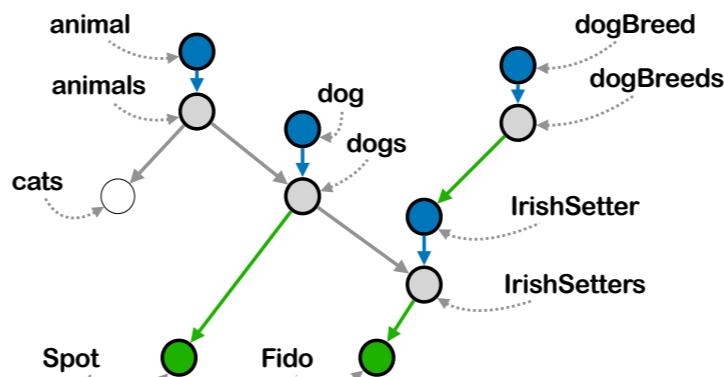
- name
- owner
- breed

add subsets, if desired:

- cats, dogs are subsets of animals

add relationships between concepts:

- IrishSetter is an example of a dogBreed
- IrishSetters is a subset of dogs
- dogs is a subset of animals

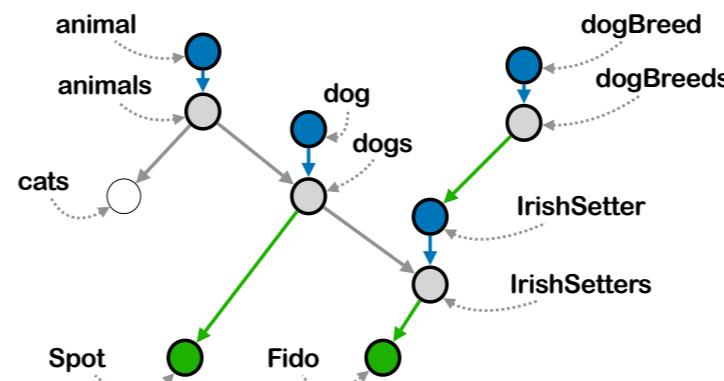


So now if I clean this up a little bit, and take all the information on the left panel and display it a little more compactly,

Conceptualization: Creation of a Concept Graph

animal

- animal
 - cats
- dog
 - Spot
 - name
 - owner
 - breed
- dogBreed
- IrishSetter
 - Fido



We have a summary of the concept graph which was created by the user. The graphical display on the right is something that some users might benefit from looking at; others may find all that information bewildering and choose not to use it. But a talented designer could probably figure out various ways to display the concept graph in an optimized manner.

Conceptualization: Creation of a Concept Graph

```
animal
• animal
  ○ cats
• dog
  ➔ Spot
  ○ name
  ○ owner
  ○ breed
• dogBreed
• IrishSetter
  ➔ Fido
```

The process of conceptualization can be applied to any app, provided the app has a database, which of course means: all of them.

Conceptualization: Creation of a Concept Graph

- | animal | dTwitter |
|--|---|
| <ul style="list-style-type: none">• animal<ul style="list-style-type: none">◦ cats• dog<ul style="list-style-type: none">→ Spot◊ name◊ owner◊ breed• dogBreed• IrishSetter<ul style="list-style-type: none">→ Fido | <ul style="list-style-type: none">• user<ul style="list-style-type: none">→ Alice→ Bob• dTweet<ul style="list-style-type: none">→ “gm”• like<ul style="list-style-type: none">→ Alice likes Bob’s “gm” dTweet◊ authorOfLike◊ authorOfTweet<ul style="list-style-type: none">◊ dTweet |

Here, for example, would be the very beginning stages of conceptualization of a database for a decentralized Twitter; starting with: a concept for users, a concept for tweets, a concept for likes. Maybe a concept for downvotes if you want that sort of thing.

Conceptualization: Creation of a Concept Graph

animal	dTwitter	Grapevine
<ul style="list-style-type: none">• animal<ul style="list-style-type: none">◦ cats• dog<ul style="list-style-type: none">→ Spot♦ name♦ owner♦ breed• dogBreed• IrishSetter<ul style="list-style-type: none">→ Fido	<ul style="list-style-type: none">• user<ul style="list-style-type: none">→ Alice→ Bob• dTweet<ul style="list-style-type: none">→ "gm"• like<ul style="list-style-type: none">→ Alice likes Bob's "gm" dTweet♦ authorOfLike♦ authorOfTweet♦ dTweet	<ul style="list-style-type: none">• rating<ul style="list-style-type: none">◦ ofUsers◦ ofTweets• ratingTemplate<ul style="list-style-type: none">◦ 5star→ TweetQuality◦ flag<ul style="list-style-type: none">→ flagAsSpam

The Grapevine app can be conceptualized: a concept for rating, which would be used to organize all of the individual ratings, and a concept for Rating Template would be starters. Conceptualization of Rating Templates would be useful if you want users to be able to create new ratingTemplates. For example, on a decentralized Twitter platform, suppose a user wants to flag users as trolls, or spam, or more specifically fakeETHGiveawaySpam, then the user could create an entry into the concept for Rating Template that serves that purpose.

Conceptualization: Creation of a Concept Graph

animal	dTwitter	Grapevine	conceptGraph
<ul style="list-style-type: none"> • animal <ul style="list-style-type: none"> ◦ cats • dog <ul style="list-style-type: none"> → Spot ◊ name ◊ owner ◊ breed • dogBreed • IrishSetter <ul style="list-style-type: none"> → Fido 	<ul style="list-style-type: none"> • user <ul style="list-style-type: none"> → Alice → Bob • dTweet <ul style="list-style-type: none"> → “gm” • like <ul style="list-style-type: none"> → Alice likes Bob’s “gm” dTweet ◊ authorOfLike ◊ authorOfTweet ◊ dTweet 	<ul style="list-style-type: none"> • rating <ul style="list-style-type: none"> ◦ ofUsers ◦ ofTweets • ratingTemplate <ul style="list-style-type: none"> ◦ 5star → TweetQuality ◦ flag <ul style="list-style-type: none"> → flagAsSpam 	<ul style="list-style-type: none"> • nodeType <ul style="list-style-type: none"> → JSONSchema → concept → superset → subset → element • relationshipType <ul style="list-style-type: none"> → yellow arrow → blue arrow → grey arrow → green arrow

Indeed, even the Concept Graph itself can be conceptualized. There would be one concept that corresponds to the different types of nodes that you have seen in many of the slides of this talk, and another concept that would correspond to the 4 different types of node-to-node relationships that make up a Loki Pathway.

[end 13]

14

The Concept Graph App

Front end

[start 14]

I've shown you a little bit about how the front end works.

The Concept Graph App

Front end

- **easy, intuitive UI for management of concept graph**

It will provide an easy, intuitive UI for the user to manage the concept graph.

The Concept Graph App

Front end

- easy, intuitive UI for management of concept graph
- allow for user-friendly visual depiction of the concept graph

And it will include visualization tools for the various graphs.

The Concept Graph App

Back end

What about the back end?

The Concept Graph App

Back end

- **create and maintain individual JSON files**

It will create and maintain individual JSON files

The Concept Graph App

Back end

- create and maintain individual JSON files
- create and maintain the graph database

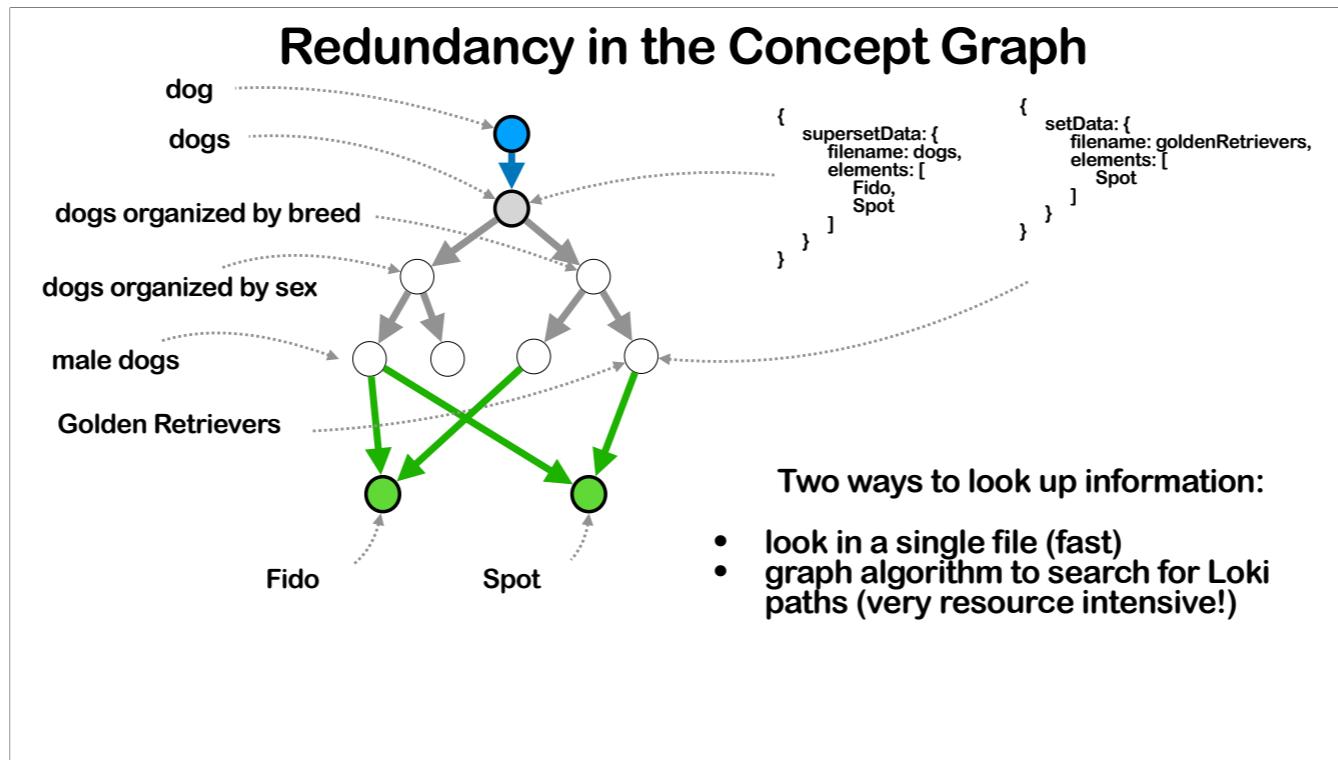
Create and maintain the graph database, which means the relationships between all of the various nodes. Some of those nodes and relationships, the user may be aware of; others, possibly not.

The Concept Graph App

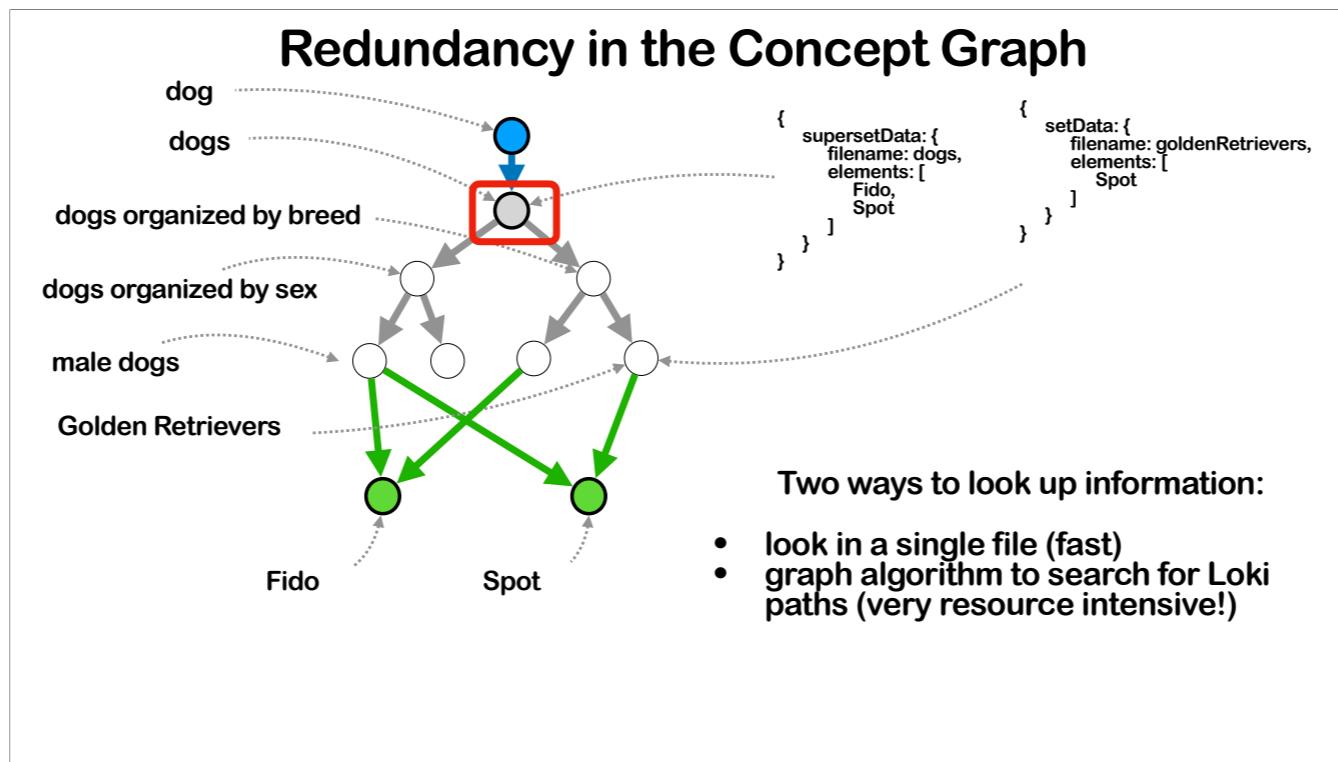
Back end

- create and maintain individual JSON files
- create and maintain the graph database
- transfer information stored in the topology of the graph into the appropriate JSON files (set nodes)

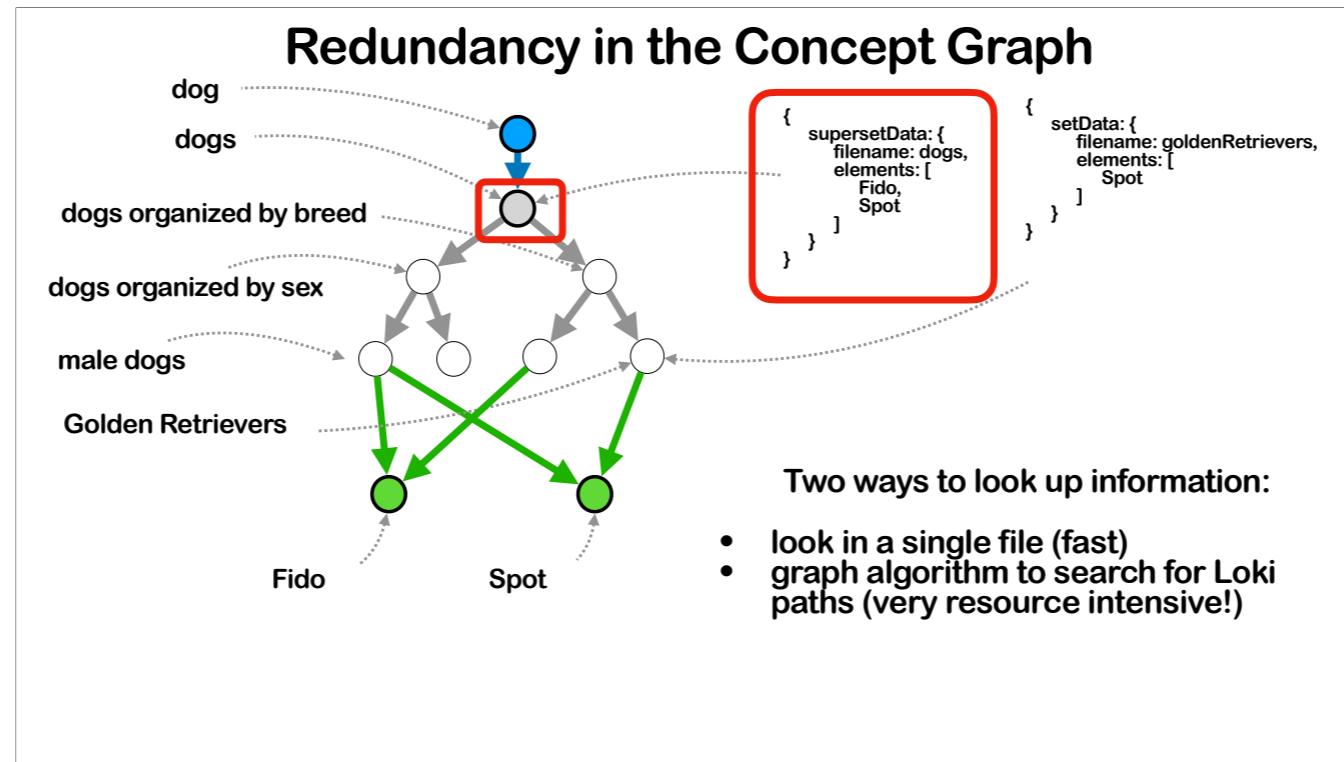
Part of maintaining JSON files will be to take information that is stored in the topology of the graph itself and use that information to modify individual JSON files. In some cases, the information flow might go the other way around.



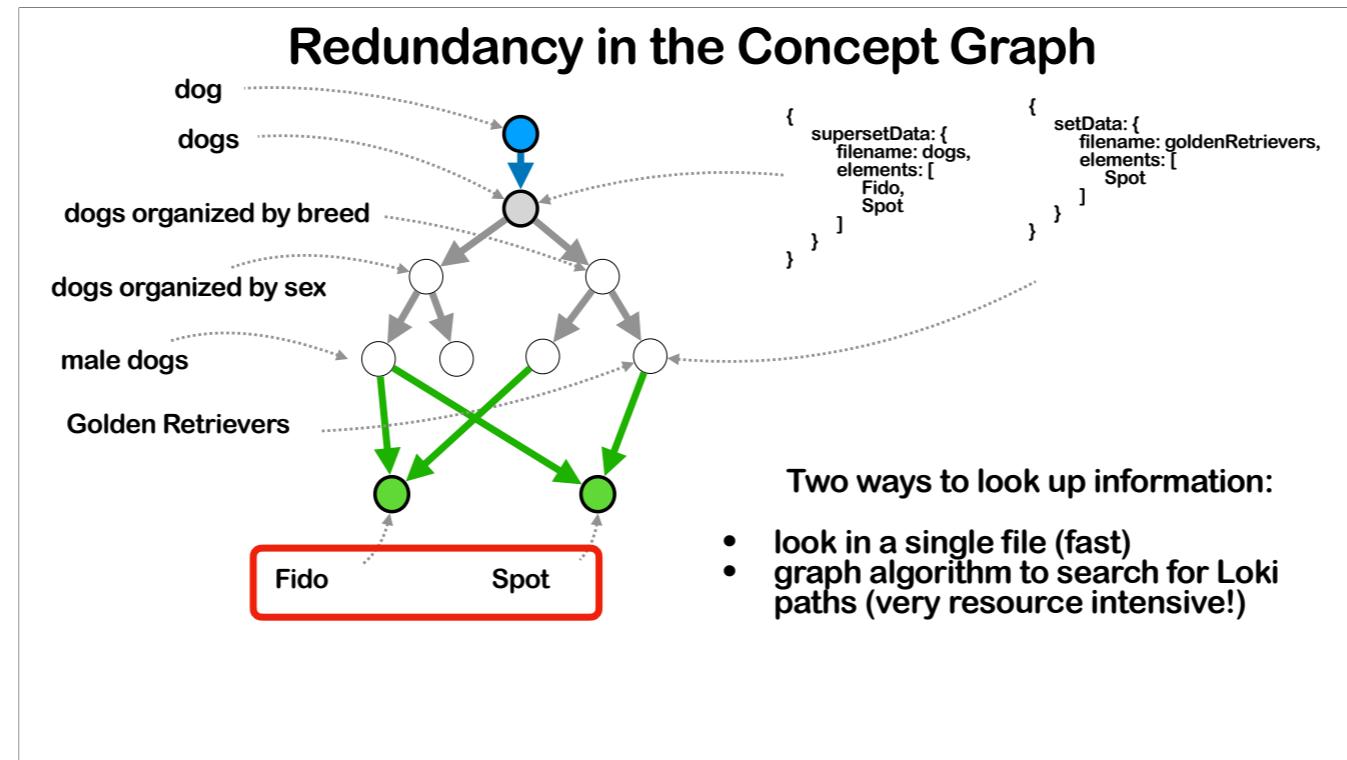
Let me give you an example of some of that information flow, from the graph into a file. Here is the concept for dog.



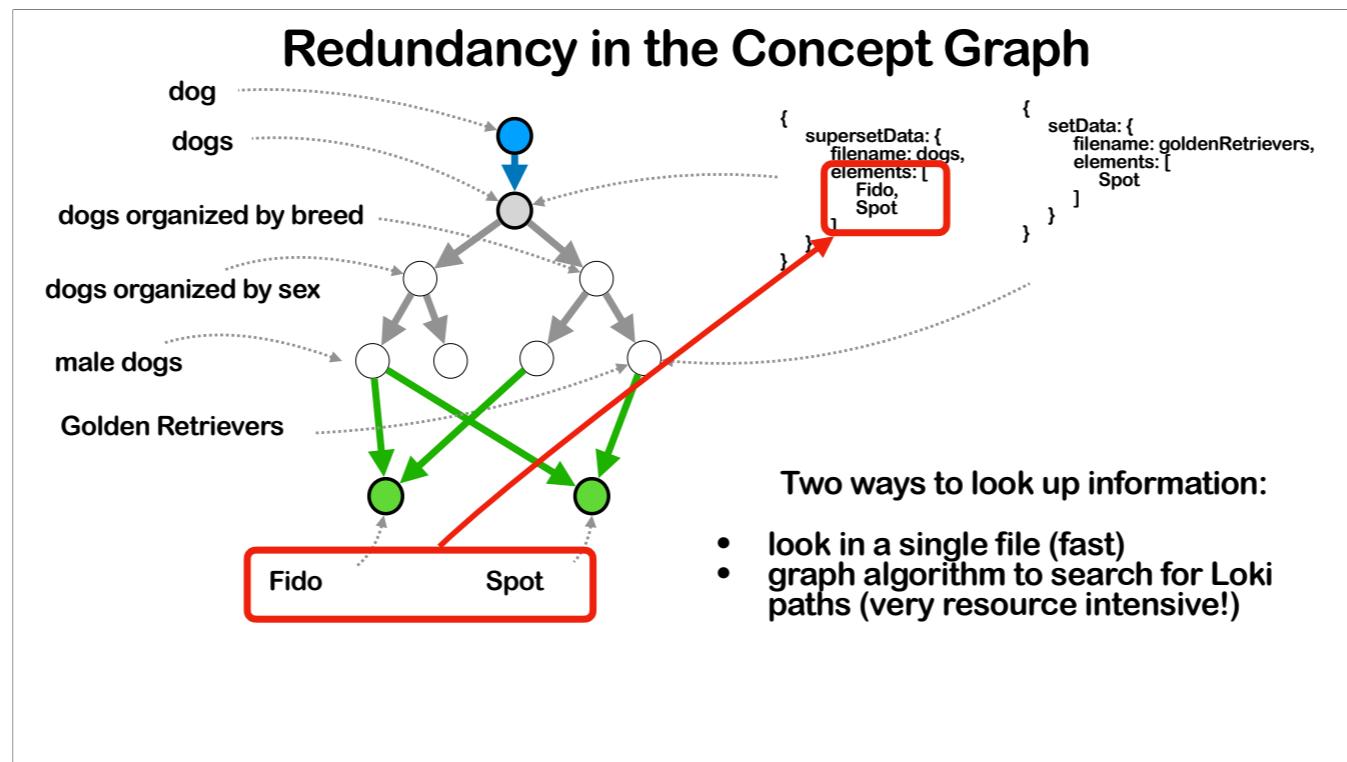
And here is the node for the superset of all dogs,



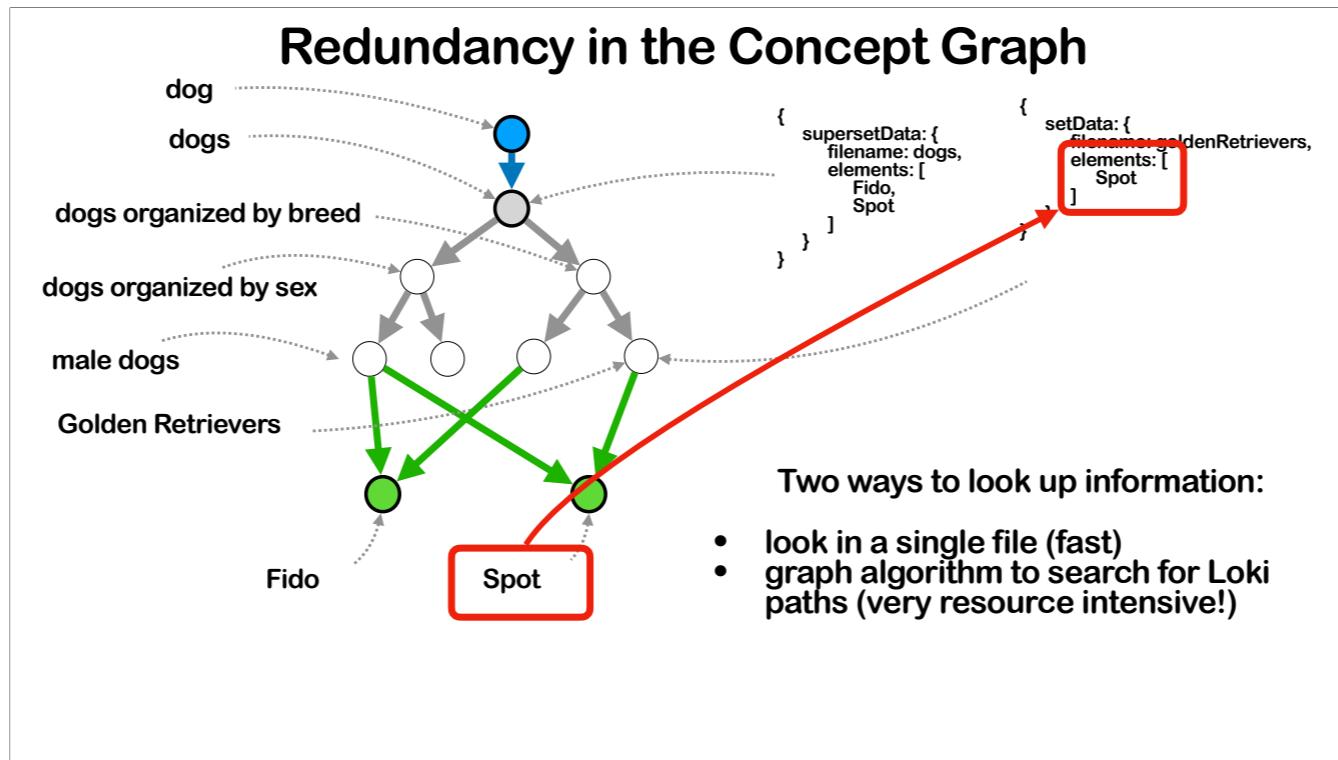
the JSON file of which is also shown.



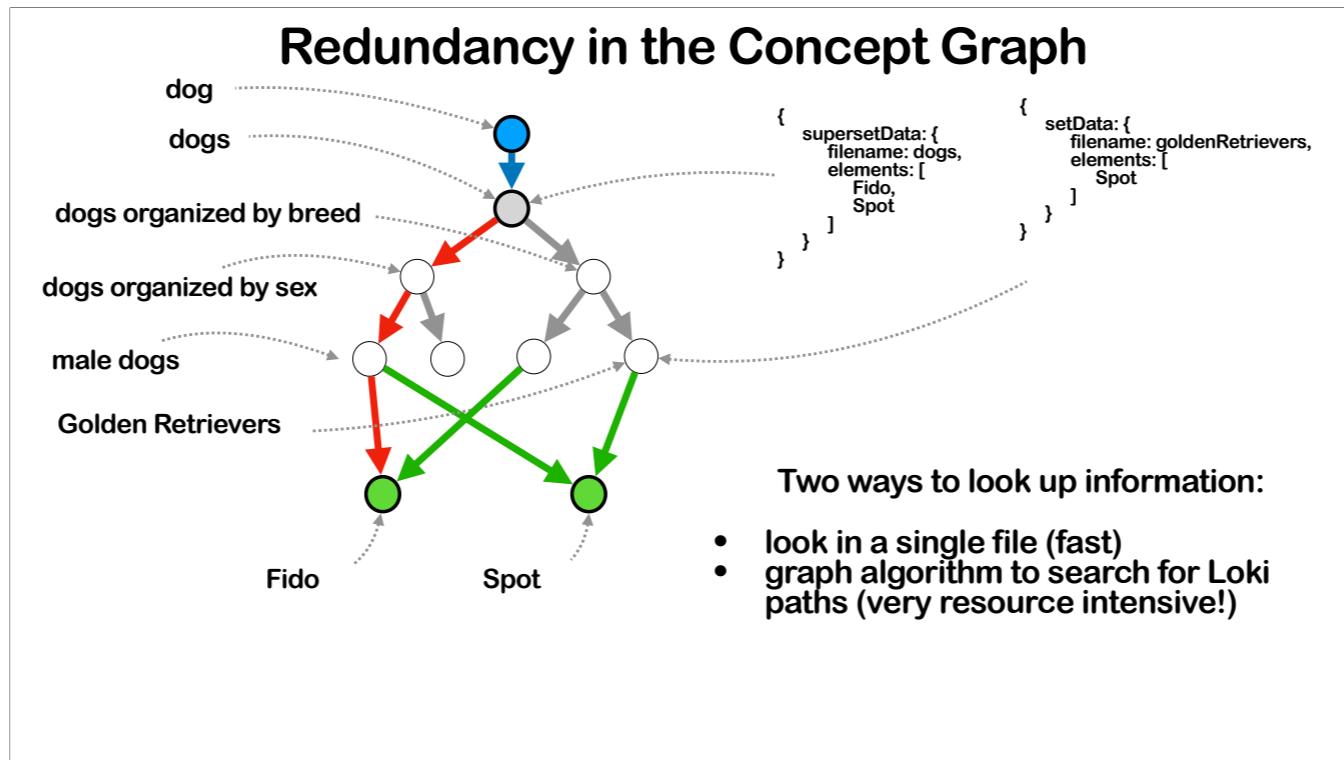
This file maintains a list of dogs that belong to that set; in this case, Fido and Spot.



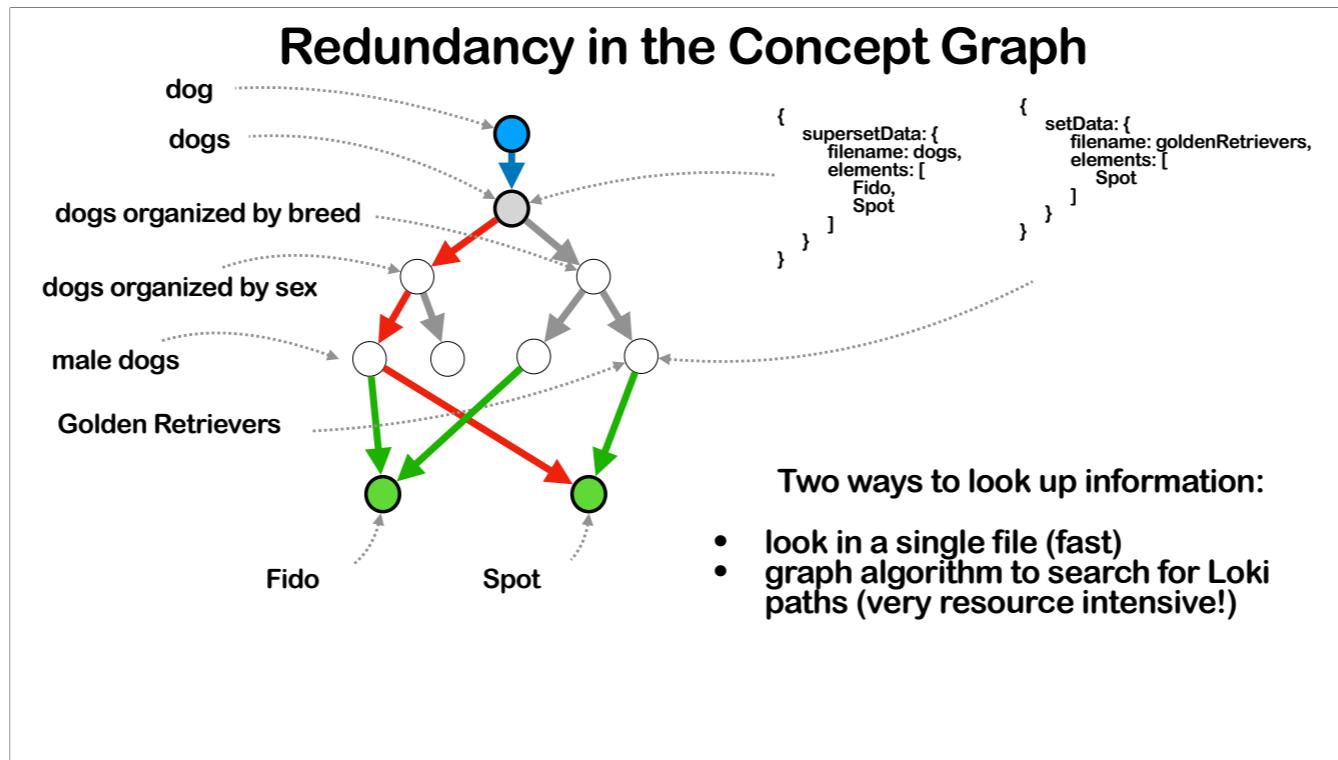
appear as elements of an array within the superset JSON file.



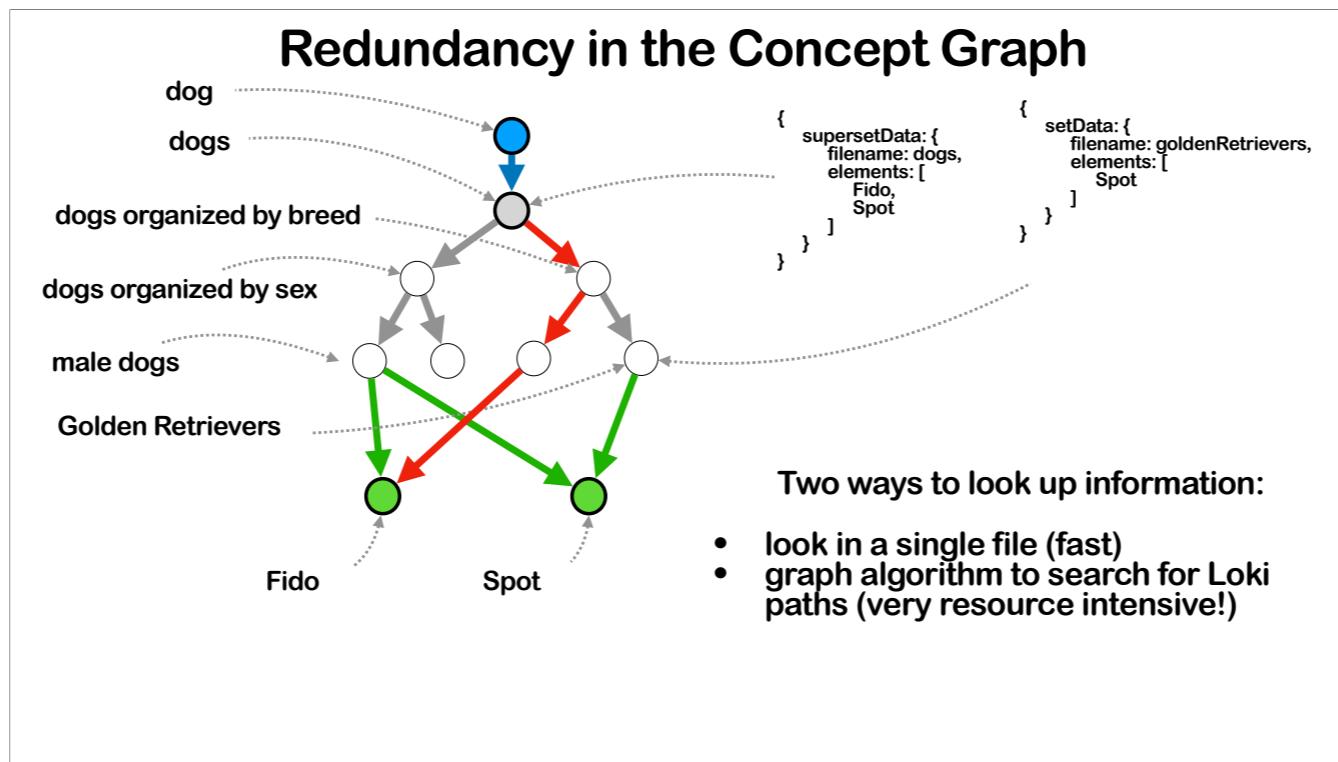
Likewise the file corresponding to the set of all golden retrievers; in this case, Spot. How does that happen?



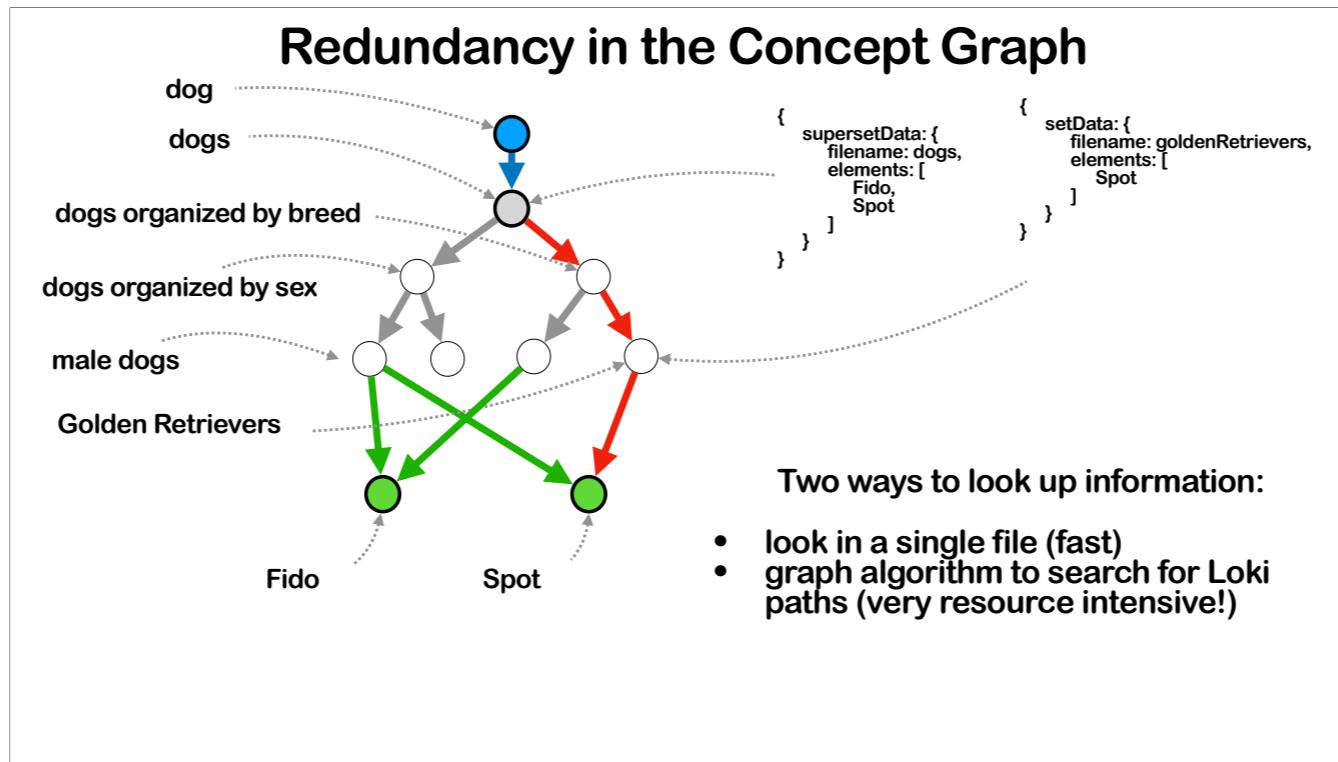
The app makes use of Loki Pathways. Essentially it explores all Loki Pathways that pass through the relevant set or superset node, on their way to an element. Here's one, in red.



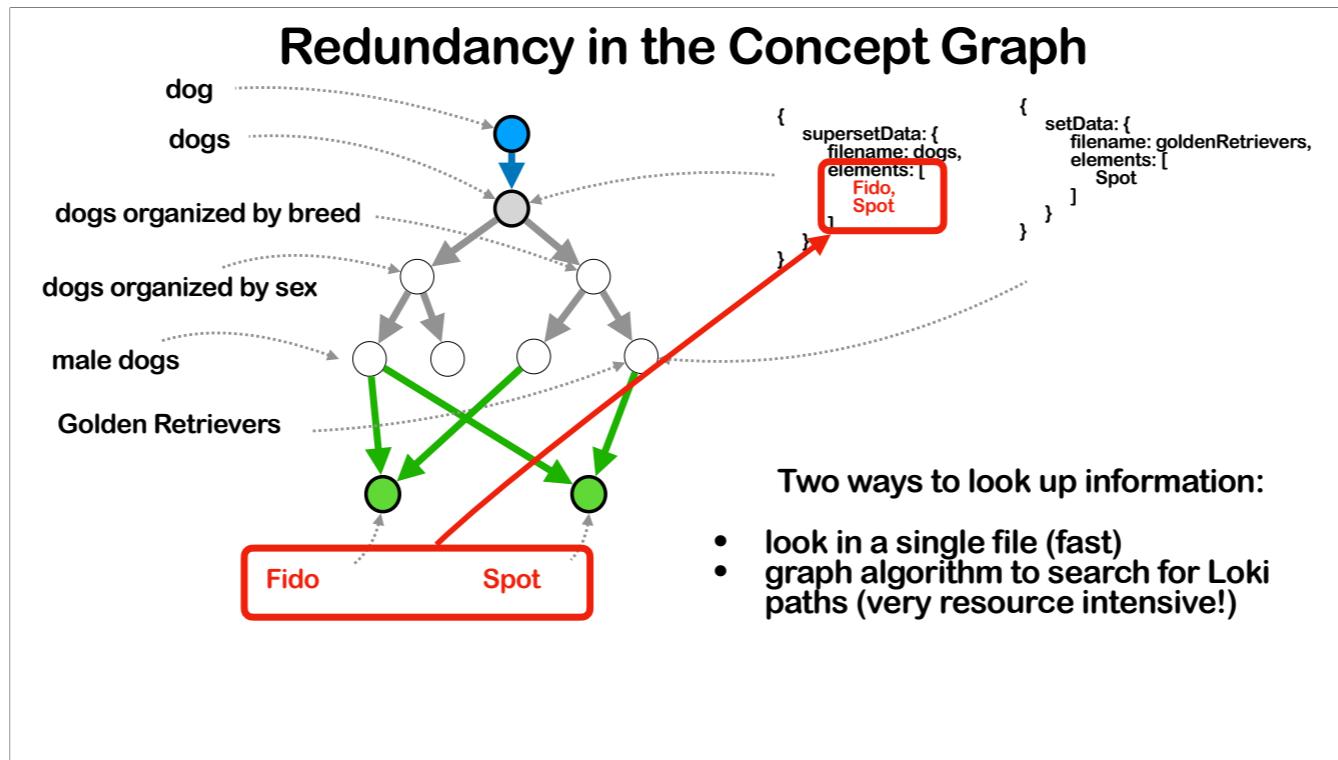
Here's one.



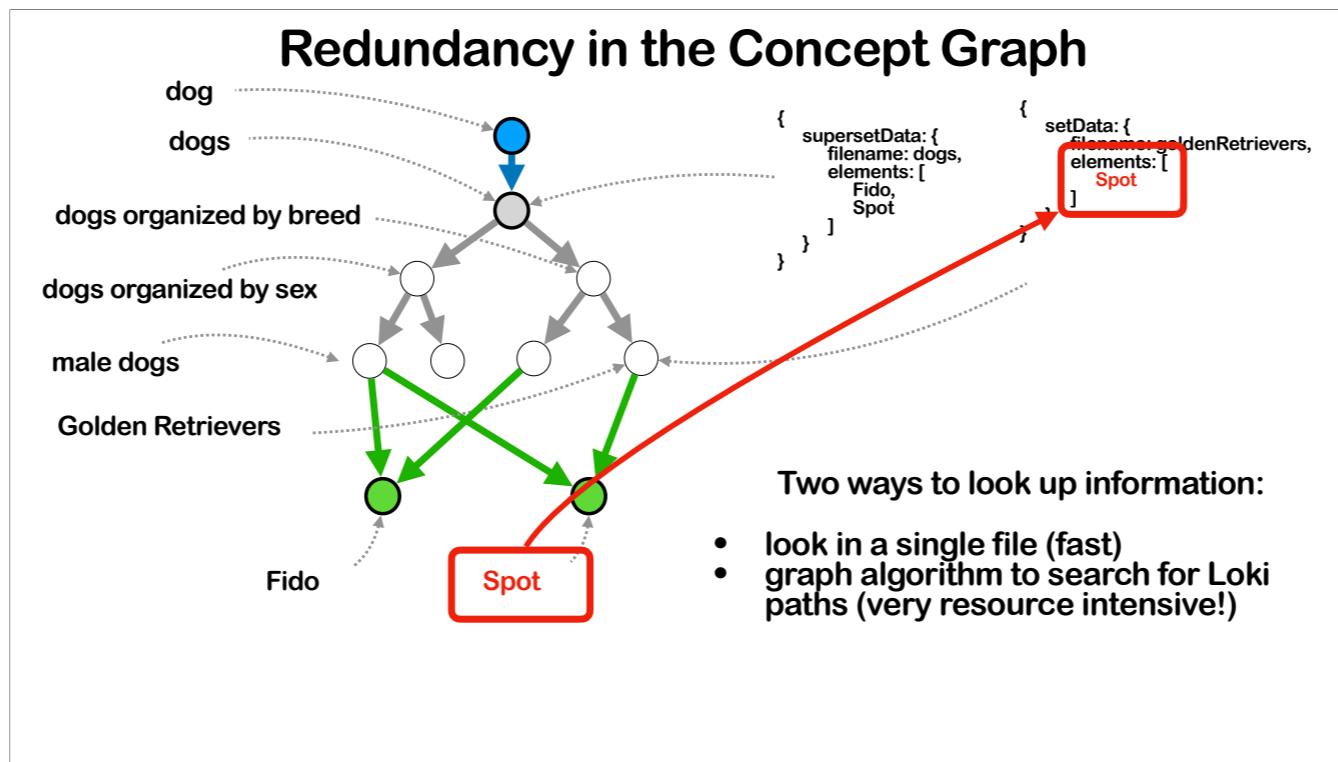
here's one.



and here's one.

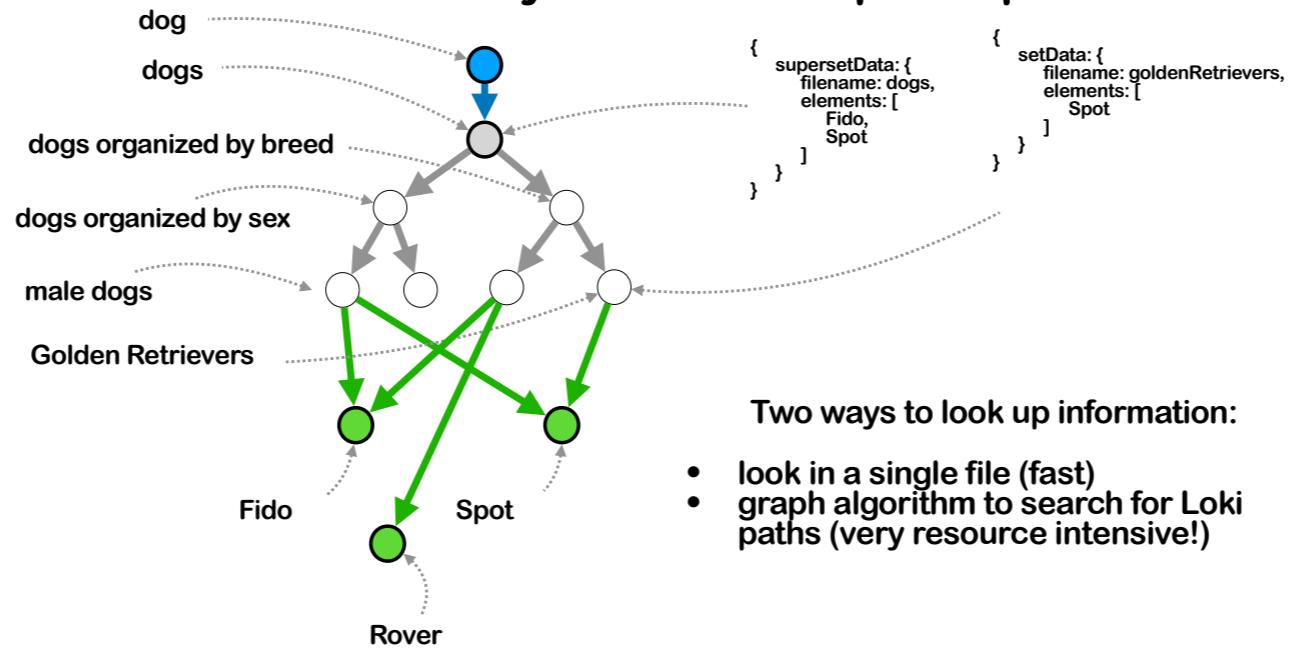


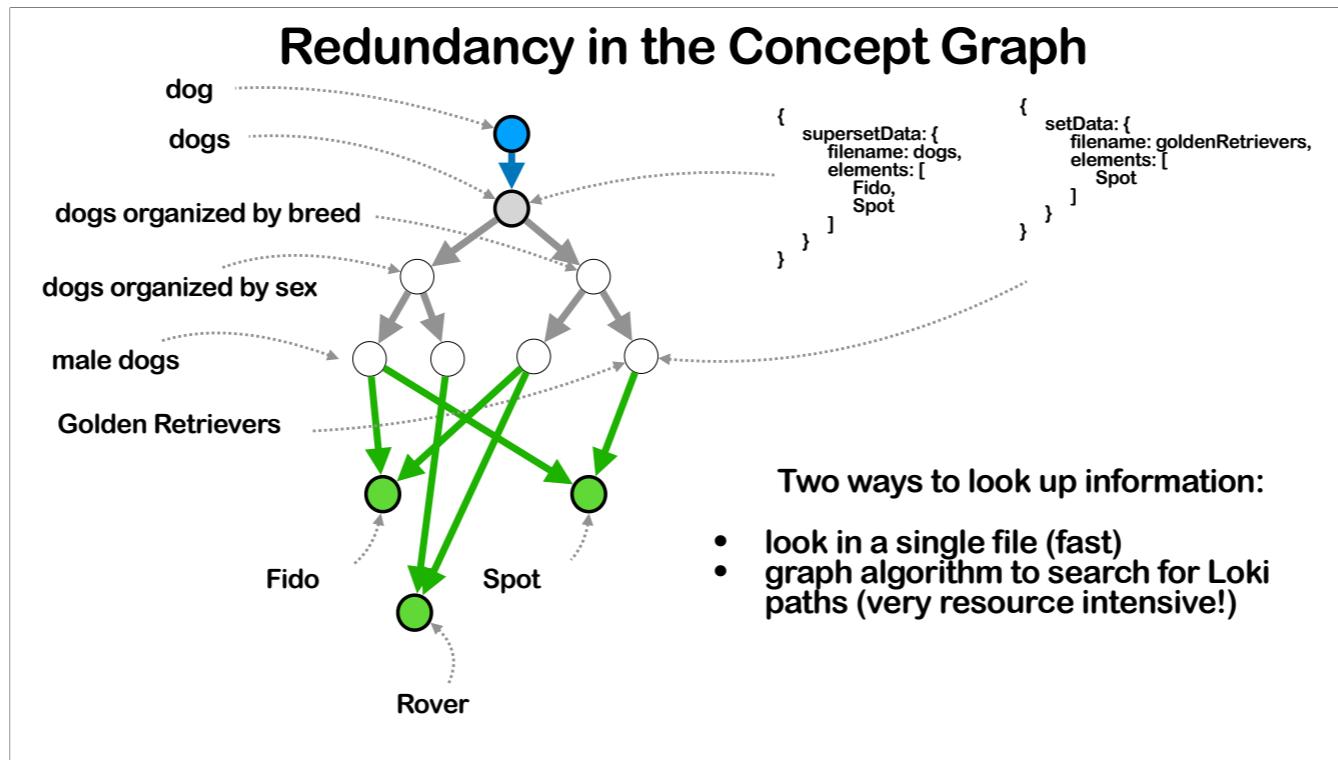
As a result, Fido and Spot get added to this file, as shown.



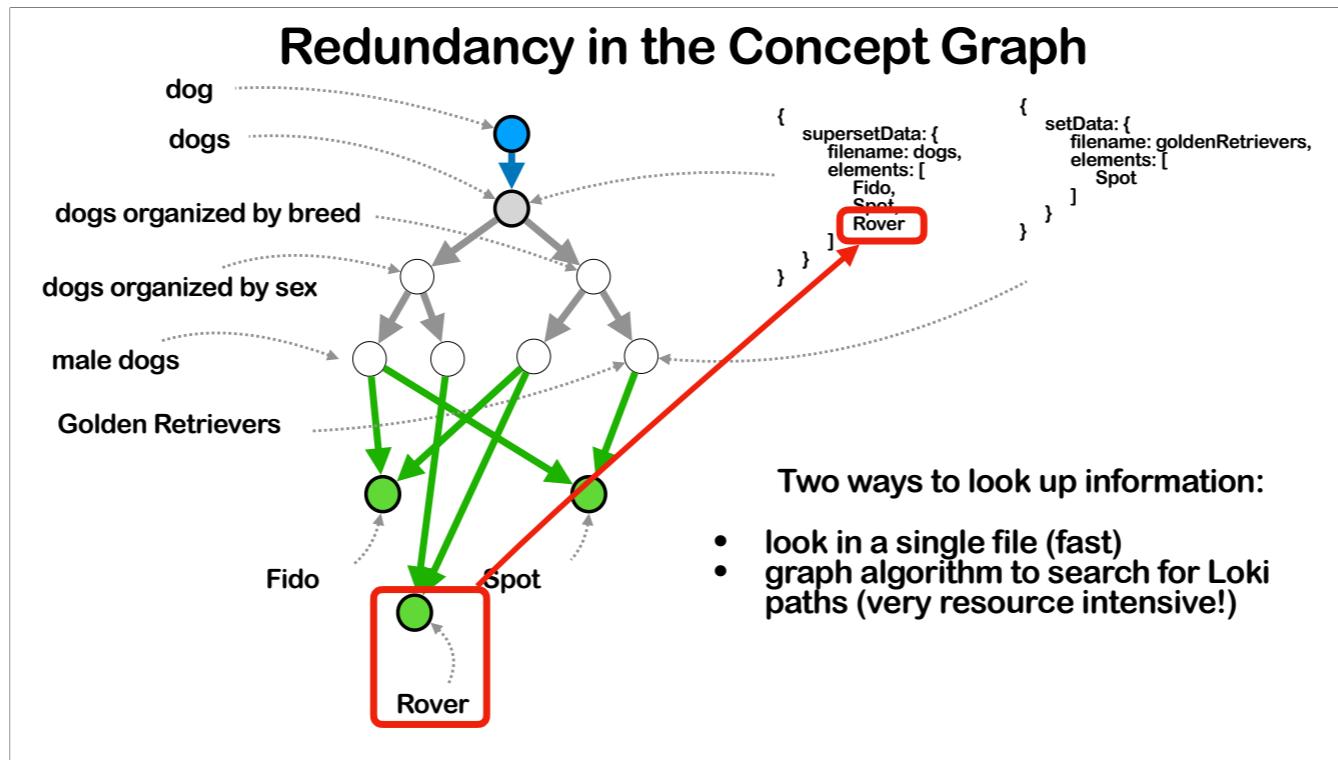
Likewise, Spot, but not Fido, gets added to the set of Golden Retrievers.

Redundancy in the Concept Graph

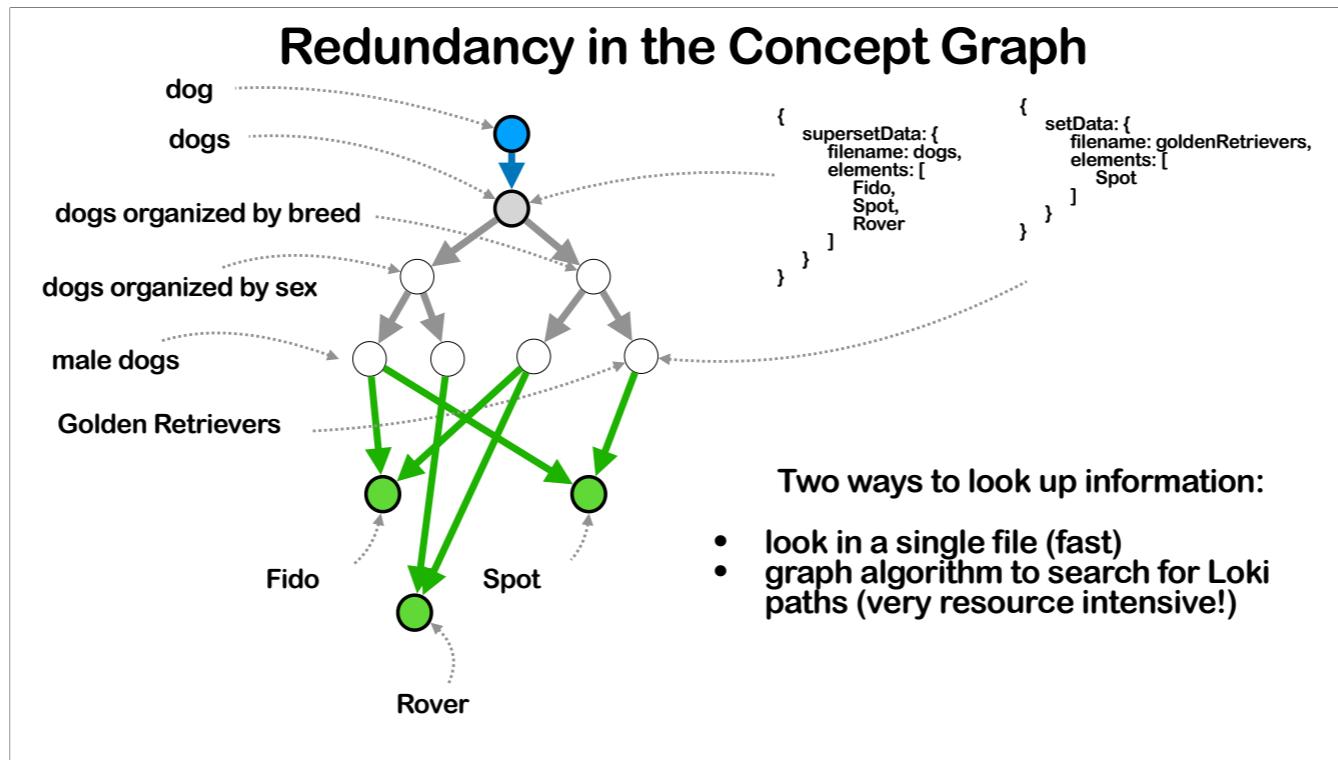




Now if an additional element gets added to your database, then in the background, the CG app will update all relevant files accordingly.



Like in this case, adding Rover to the superset file, but not the file for the set of Golden Retrievers.

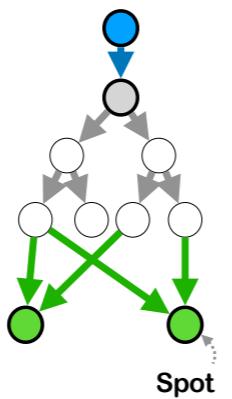


One of the things you'll notice is that the information in your database as well as the complexity of your database can be updated gradually. There is more flexibility than would be the case for a relational database, which tends to be more brittle.

[end 14]

15

extracting information FROM the graph topology

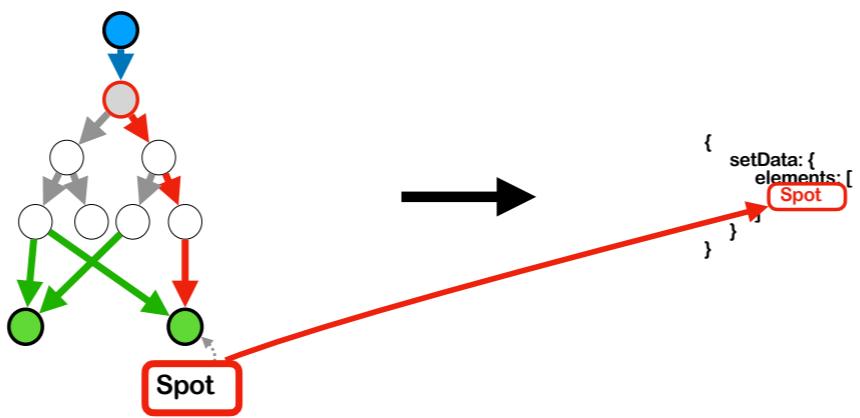


```
{  
    setData: {  
        elements: [  
            ]  
    }  
}
```

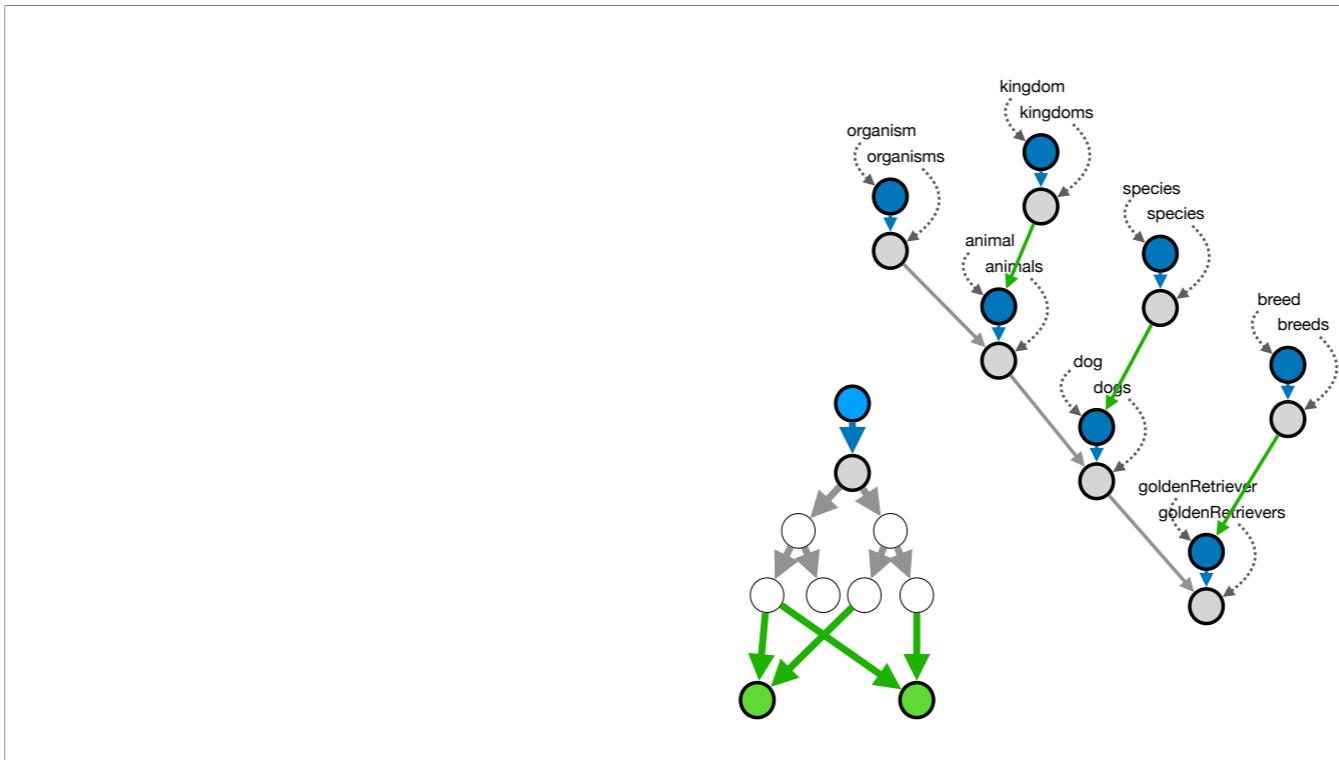
[start 15]

I've shown instances of information flowing from the topology of the graph on the left

extracting information FROM the graph topology

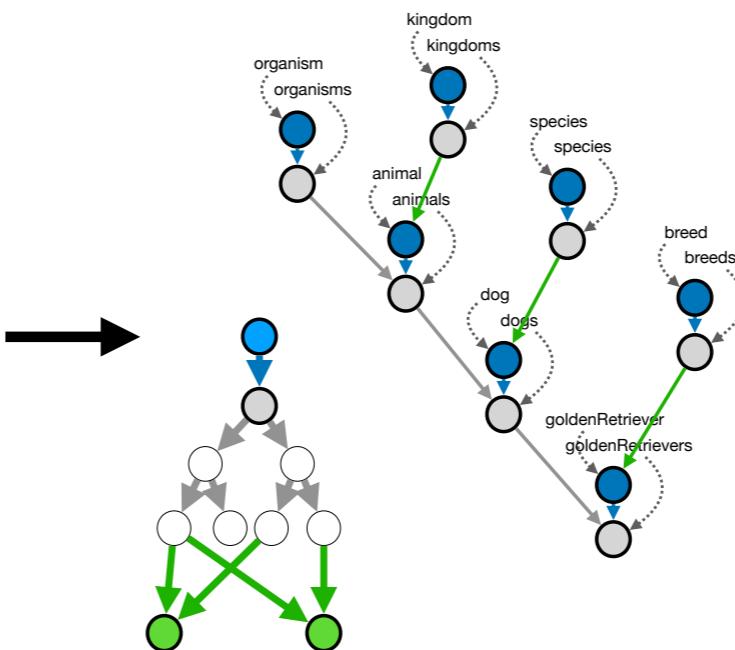


into a particular file on the right.



There are also instances where information will flow in the opposite direction:

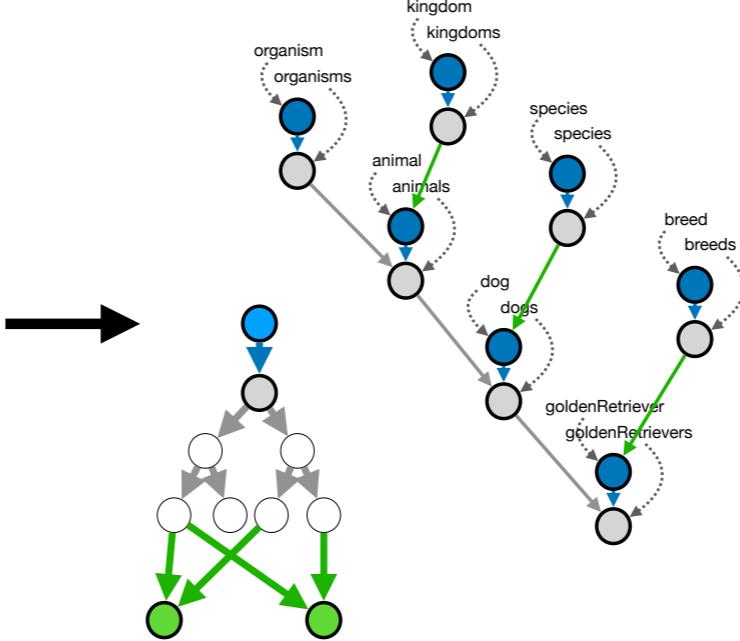
- files
- user input



from a file, or from direct input from the user, to be stored into the topology of the graph.

topological encoding of information INTO the graph

- files
- user input



topological encoding of information INTO the graph

I will show you some examples.

topological encoding of information INTO the graph front end

I'll show you what happens on the front end

topological encoding of information INTO the graph

front end

back end

and on the back end of the app.

topological encoding of information INTO the graph

front end

back end

 concept list

The first task for a user who is creating a concept graph is to create some new concepts.

topological encoding of information INTO the graph

front end

back end

 concept list

- dog

Let's start with dog.

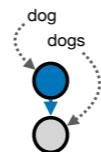
topological encoding of information INTO the graph

front end

 concept list

- dog

back end



On the back end, the app makes two connected files, dog and dogs,

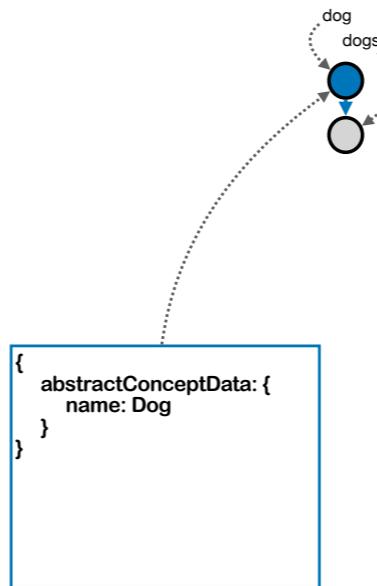
topological encoding of information INTO the graph

front end

back end

 concept list

- dog



and creates the associated JSON files for dog

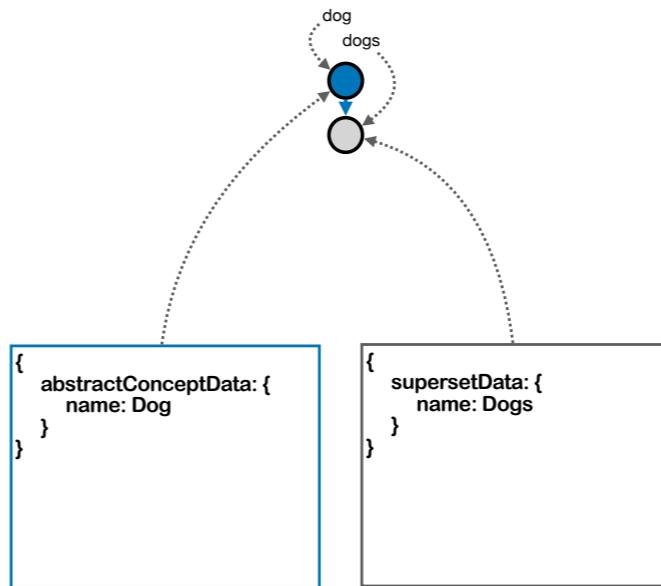
topological encoding of information INTO the graph

front end

back end

 concept list

- dog



and dogs, which would look something like this.

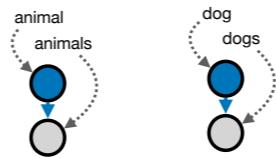
topological encoding of information INTO the graph

front end

back end

 concept list

- dog
- animal



likewise for animal

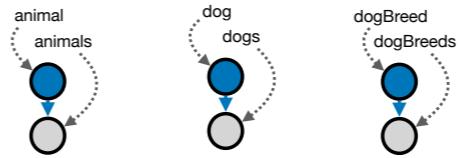
topological encoding of information INTO the graph

front end

concept list

- dog
- animal
- dogBreed

back end



and dog breed

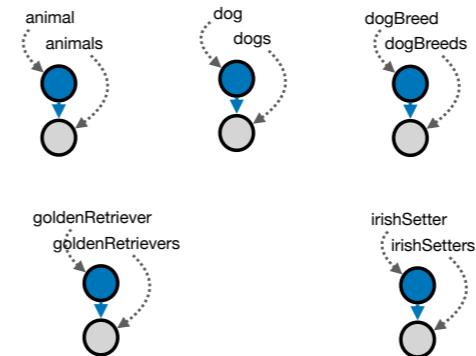
topological encoding of information INTO the graph

front end

concept list

- dog
- animal
- dogBreed
- goldenRetriever
- irishSetter

back end

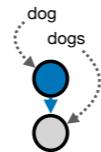


and gR and iS

topological encoding of information INTO the graph

front end

back end



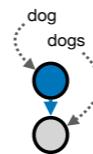
Now, getting back to the concept for dog

topological encoding of information INTO the graph

front end

+ elements of dog

back end



we want to flesh out the concept by adding some elements; meaning, some examples of the concept.

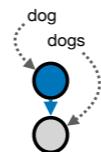
topological encoding of information INTO the graph

front end

+ elements of dog

→ Spot

back end



like spot

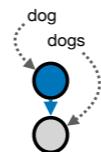
topological encoding of information INTO the graph

front end

⊕ elements of dog

→ Spot
→ Fido

back end



and Fido

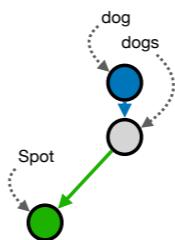
topological encoding of information INTO the graph

front end

⊕ elements of dog

→ Spot
→ Fido

back end



Underneath the hood, the CG app adds Spot,

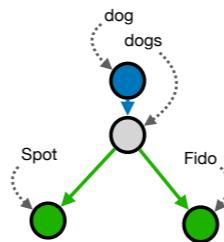
topological encoding of information INTO the graph

front end

⊕ elements of dog

→ Spot
→ Fido

back end



and Fido, both as elements of the superset of all dogs, since we don't have any subsets yet,

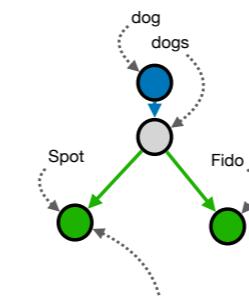
topological encoding of information INTO the graph

front end

⊕ elements of dog

→ Spot
→ Fido

back end



```
{\n    dogData {\n    }\n}
```

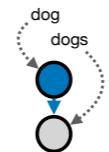
with the file for Spot looking like this. You note here that dogData is an object but is empty, and that's bc we haven't yet added any properties to the concept for dog. So that's what we do next.

topological encoding of information INTO the graph

front end

 properties for dog

back end



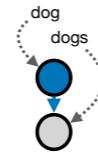
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



We add name, owner, and breed as three properties, all of which are simple strings.

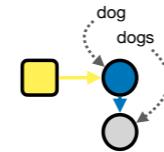
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



Now behind the scenes, a lot happens. The app creates a node for the concept's JSON Schema,

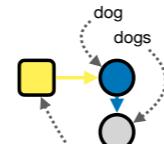
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



```
{  
  type: object,  
  properties:  
    ...  
}
```

and initializes its JSON file;

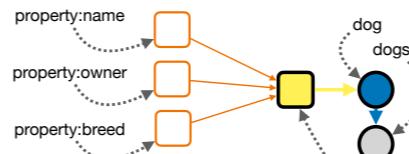
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



```
{  
  type: object,  
  properties {  
    ...  
  }  
}
```

it then adds three property nodes, shown here as orange squares, corresponding to the three properties that the user just created,

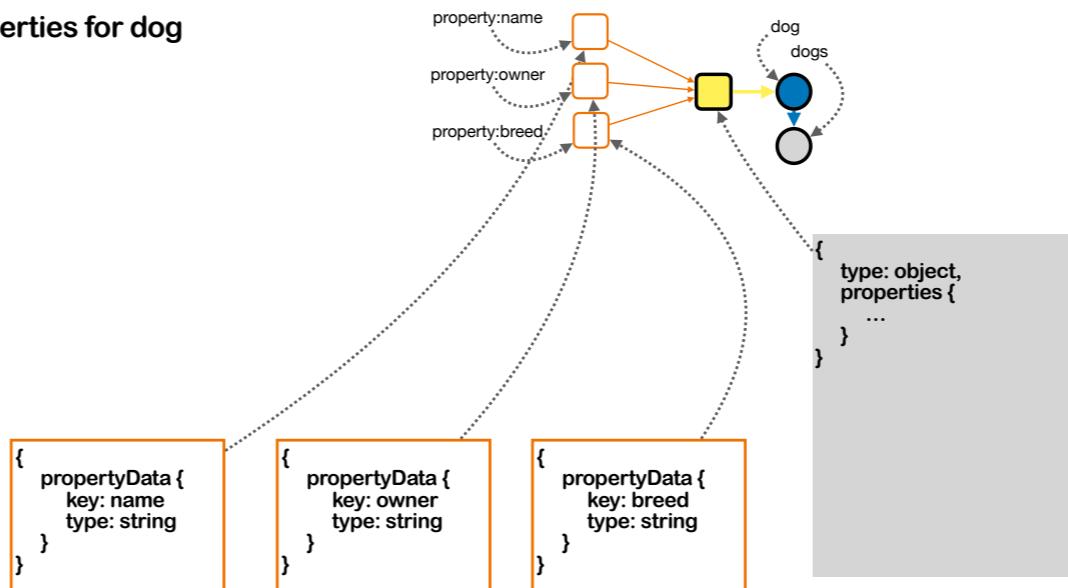
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



and each one has its own file, which acts as a record for the key and for the type of the property value, as shown.

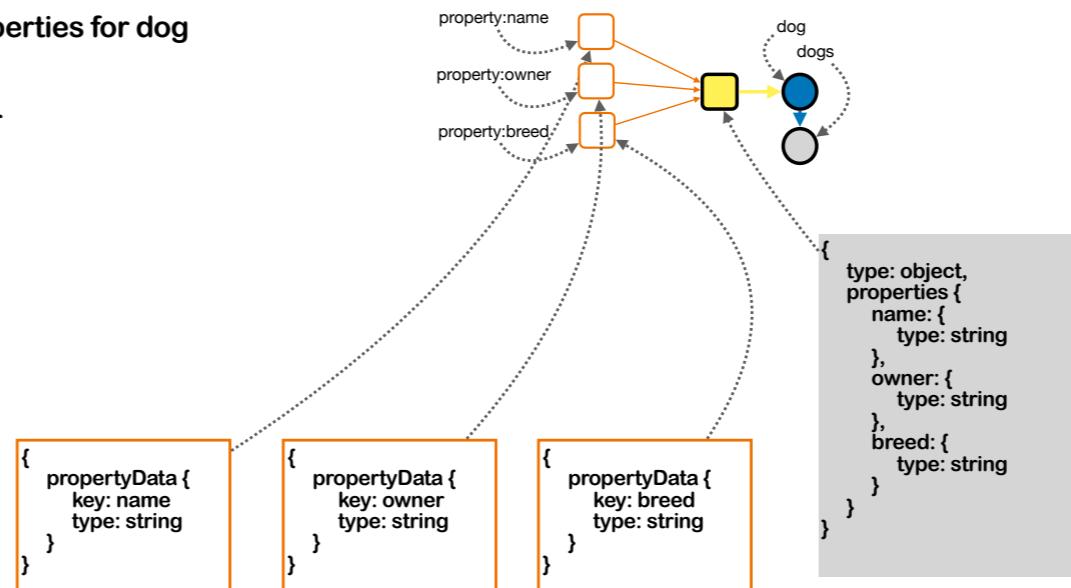
topological encoding of information INTO the graph

front end

⊕ properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



These three properties propagate automatically into the JSON Schema on the right. Basically, the orange arrows that you see in the graph, pointing towards the yellow JSON Schema node, are what trigger for that to happen.

topological encoding of information INTO the graph

front end

back end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

Now that properties have been added,

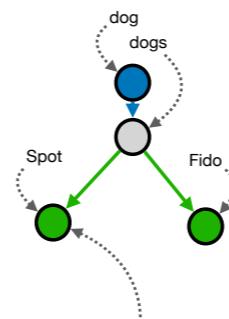
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



```
{\n    dogData {\n    }\n}
```

we can go back to the individual elements, like Spot,

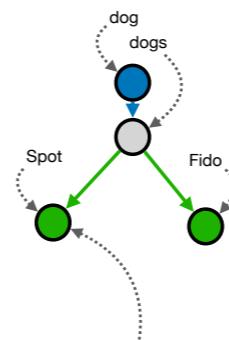
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed

back end



```
{  
  dogData {  
    name: Spot,  
    owner: Alice,  
    breed: Irish Setter  
  }  
}
```

and fill in the details, like so.

topological encoding of information INTO the graph

front end

back end



We can go through the same process for dogBreed that we did for dog.

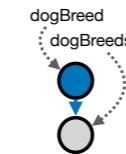
topological encoding of information INTO the graph

front end

 concept list

- dogBreed

back end



define the concept.

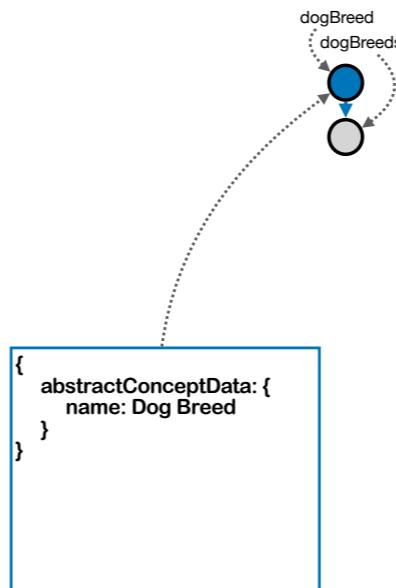
topological encoding of information INTO the graph

front end

 concept list

- dogBreed

back end



The associated files for dog breed

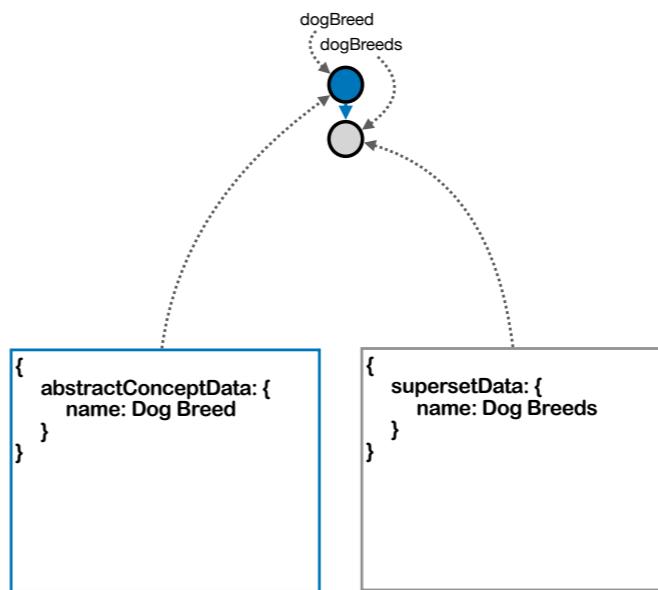
topological encoding of information INTO the graph

front end

 concept list

- dogBreed

back end



and dog breeds would look something like this.

topological encoding of information INTO the graph

front end

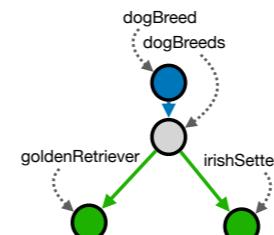
concept list

- dogBreed

elements of dogBreed

- ➡ Golden Retriever
- ➡ Irish Setter

back end



add some elements: Golden Retriever, and Irish Setter.

topological encoding of information INTO the graph

front end

[+] concept list

- dogBreed

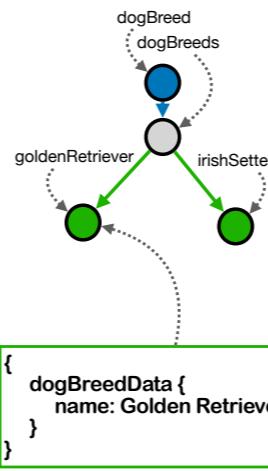
[+] elements of dogBreed

- ➡ Golden Retriever
- ➡ Irish Setter

[+] properties for dogBreed

- ❖ name

back end



add the property, name. We could add more if we wanted but name is all we will need.

topological encoding of information INTO the graph

[end 15]

16

topological encoding of information INTO the graph relationships between concepts

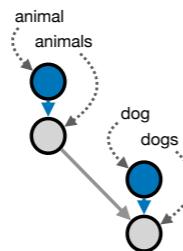
[start 16]

I'm going to speak now about one of the more complicated topics relating to the concept graph: the relationships between concepts!

**topological encoding of information INTO the graph
relationships between concepts**

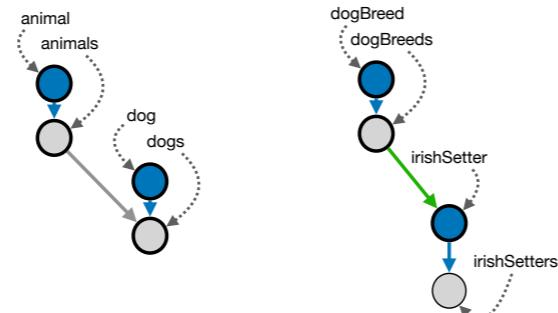
They will come in three varieties.

topological encoding of information INTO the graph relationships between concepts



In the first category: one concept can be a subset of another concept. In this case we see the grey arrow, indicating that dogs is a subset of animals.

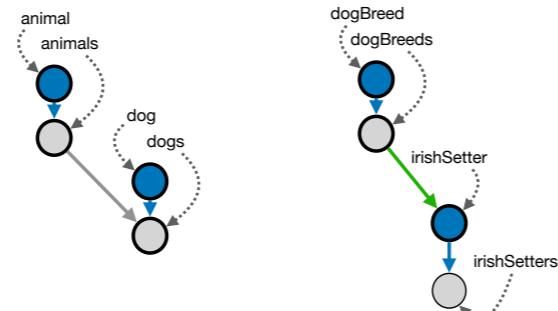
topological encoding of information INTO the graph relationships between concepts



In the second category:

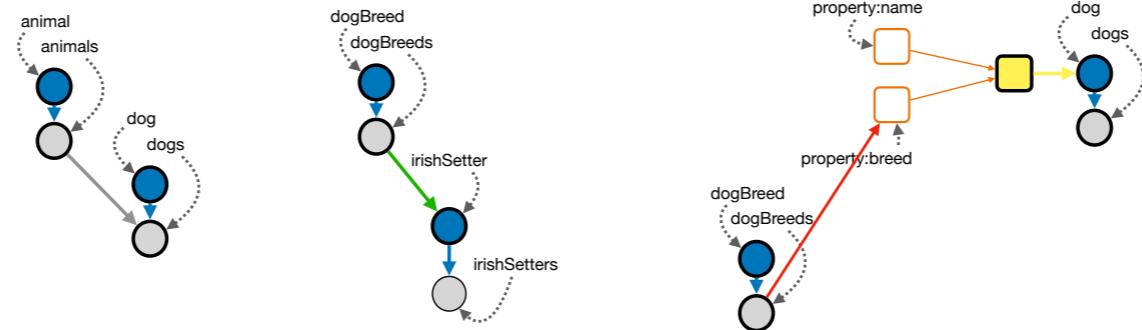
One concept can be an example of another concept. In this case we see the green arrow, indicating that the abstract concept of an Irish Setter is an example of a breed.

topological encoding of information INTO the graph relationships between concepts



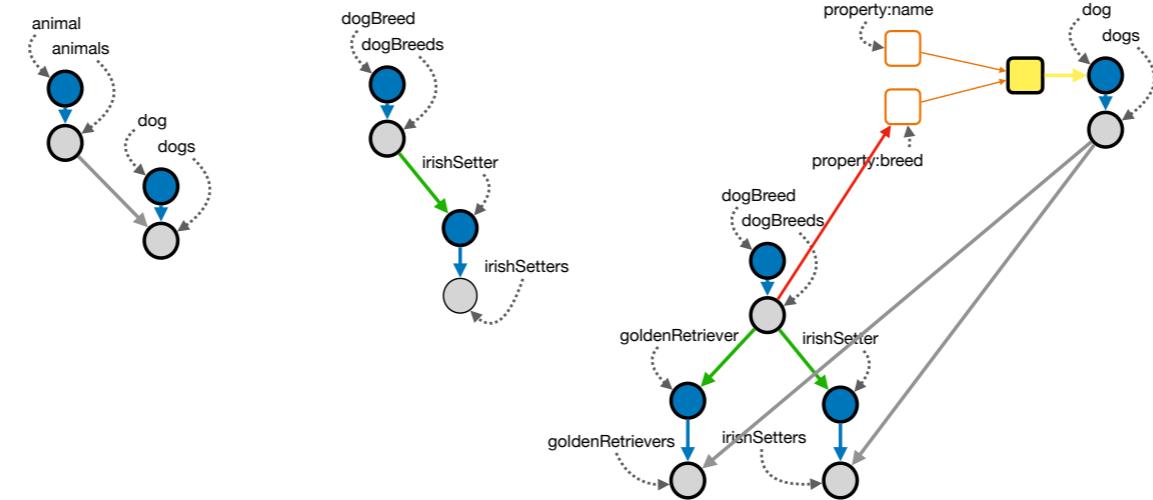
These first two categories of the relationships between concepts are the simplest two categories of such relationships, but they do demonstrate why it is necessary for any concept to be separated into the abstract idea, shown as the blue node, and the superset, shown as the grey node. If we were not to do that, it would screw up our Loki Pathways. Evidence once again that our one simple starting idea of the Loki Principle is able to guide, perhaps even to dictate, many of our design decisions.

topological encoding of information INTO the graph relationships between concepts



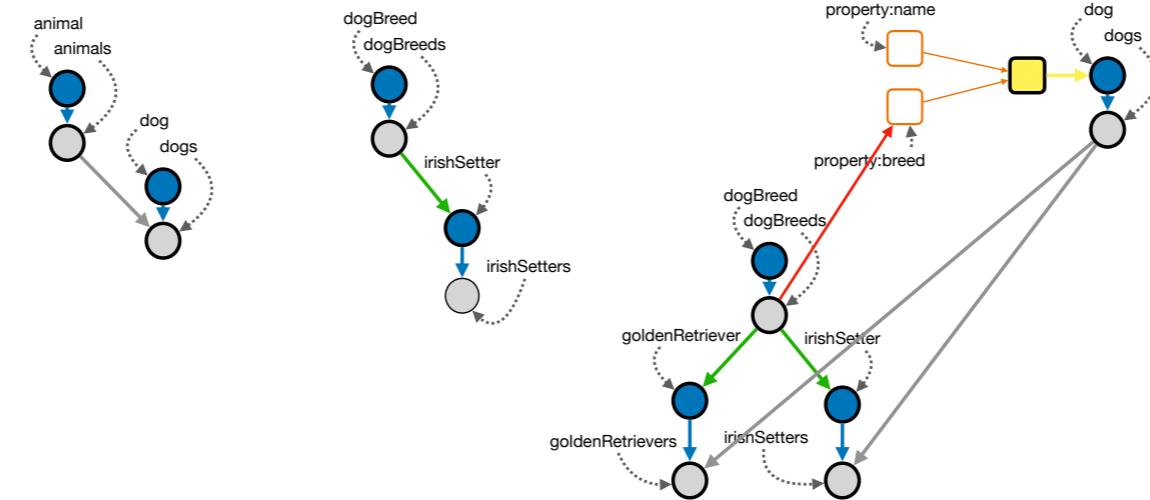
The third category is utilized when we want one concept to modify or place restrictions upon one of the properties within the Property Tree of another concept. In this example you see the red arrow indicating that the property breed, within the property tree of the concept for dogs, is not allowed to be ANY string, but instead is limited to being one of the elements of the other concept, dog breeds. This third category is much more complex than the first two.

topological encoding of information INTO the graph relationships between concepts



For one thing, it is often the case that the relationship you see as the red arrow will necessarily imply a slate of additional relationships, as you see here.

topological encoding of information INTO the graph relationships between concepts



For another thing, there are many different ways to structure a property (for example, the property type can be a string, or an array, or an object, and so on), and as such, there are many different ways that one concept can be used as input in the construction of a property. I'll be walking step by step through two such ways in this video.

topological encoding of information INTO the graph relationships between concepts

So I'll walk through step by step what happens on the front end and on the back end when a user wants to create one of these inter-concept relationships. One of the points I want to demonstrate is that there is no need for the user experience to be complicated. Everything the user does will be intuitive. There is no need for the user to have any idea what's going on under the hood.

topological encoding of information INTO the graph relationships between concepts

- one concept forms a subset of another concept

We'll start with something simple: one concept forming a subset of another concept.

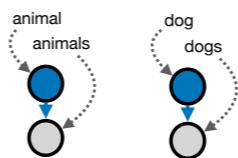
topological encoding of information INTO the graph

front end

 concept list

- dog
- animal

back end



Let's say the user creates a concept for dog and a concept for animal.

topological encoding of information INTO the graph

front end

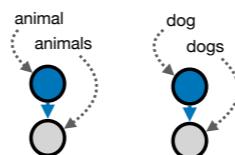
concept list

- dog
- animal

relationships list

- dogs is a subset of animals

back end



The user thinks, hmm, how are these related? Well, dogs is a subset of animals. At the front end, the user selects a few options, clicks a few buttons and creates this relationship: dogs is a subset of animals. A simple sentence, that you don't have to be a developer to understand intuitively what that means and that it is correct.

topological encoding of information INTO the graph

front end

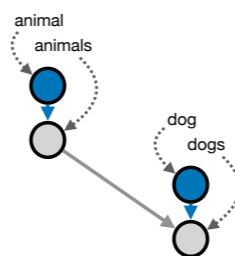
concept list

- dog
- animal

relationships list

- dogs is a subset of animals

back end



The action at the back end looks like this. Keep in mind: the user may or may not want to dive in to what the graph looks like. The user may or may not keep track of the fact that a grey arrow in a Concept Graph app from animals to dogs means that dogs is a “subset of” animals.

topological encoding of information INTO the graph relationships between concepts

- one concept forms a subset of another concept

OK, we have done subsets.

topological encoding of information INTO the graph relationships between concepts

- one concept forms a subset of another concept
- one concept forms an element of another concept

Another way that concepts can relate to one another is that one concept can form an element of another concept.

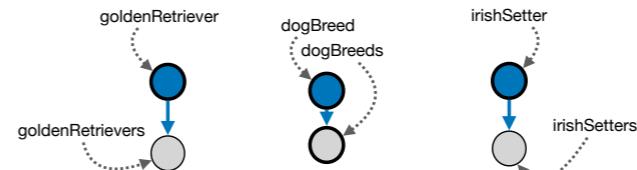
topological encoding of information INTO the graph

front end

 concept list

- dogBreed
- goldenRetriever
- irishSetter

back end



Let's start with the concepts for dogBreed, gR, and iS.

topological encoding of information INTO the graph

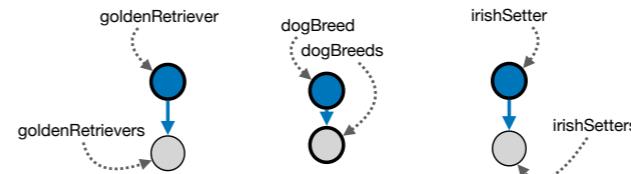
front end

+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

back end



The user thinks: hmmm, how are these three concepts related to one another?

Well, iS and gR are examples of breeds.

So he clicks a few buttons and selects that relationship

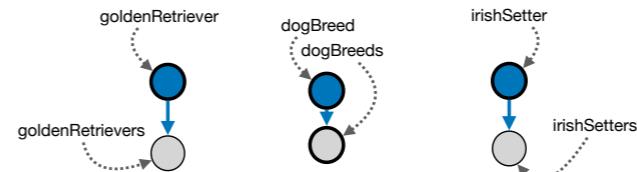
topological encoding of information INTO the graph

front end

+ concept list

- dogBreed
- goldenRetriever
- irishSetter

back end



+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

and enters it on the front end.

topological encoding of information INTO the graph

front end

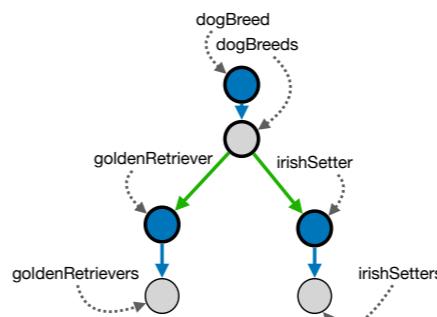
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



On the back end, the CG app adds the two green arrows that you see here, which indicate that gR and iS are elements of the set of dog breeds.

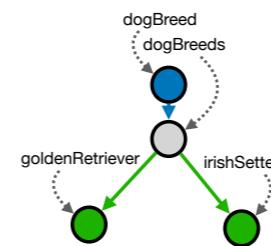
topological encoding of information INTO the graph

front end

 concept list

- dogBreed
- goldenRetriever
- irishSetter

back end



Recall that a few slides ago, gR and iR were added as dog breeds,

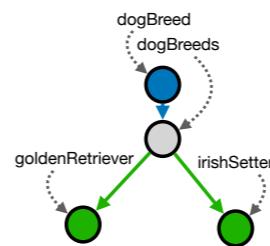
topological encoding of information INTO the graph

front end

 concept list

- dogBreed

back end



but this was done without having previously defined them as concepts.

topological encoding of information INTO the graph

front end

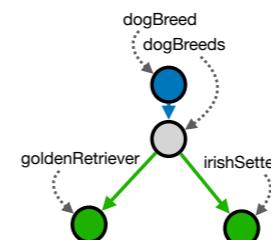
+ concept list

- dogBreed

+ elements of dogBreed

- ➡ Golden Retriever
- ➡ Irish Setter

back end



They were simply created de novo as elements of dog breed.

topological encoding of information INTO the graph

front end

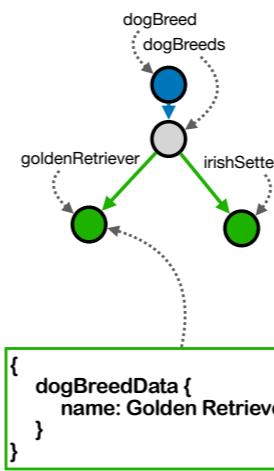
 concept list

- dogBreed

 elements of dogBreed

- ➡ Golden Retriever
- ➡ Irish Setter

back end



In that case, the file looked like this, with a property key called dogBreedData.

topological encoding of information INTO the graph

front end

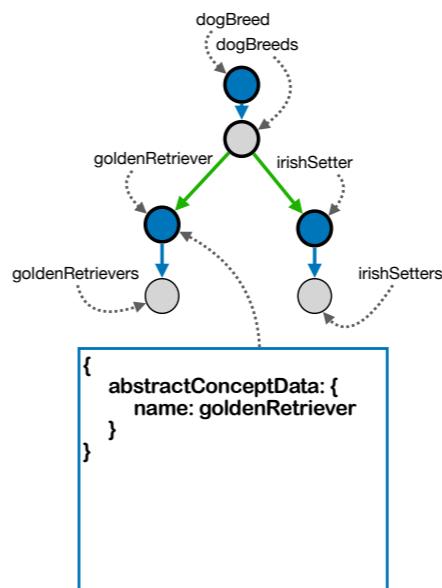
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



In this case, the original file for gR, prior to adding it under dB, looked like this, with a property key called abstractConceptData

topological encoding of information INTO the graph

front end

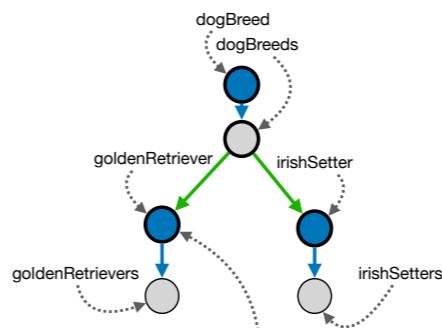
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



```
{ abstractConceptData: { name: goldenRetriever }, dogBreedData { name: Golden Retriever } }
```

But after adding it as an example of a dogBreed, it looks like this. Both properties are present:

topological encoding of information INTO the graph

front end

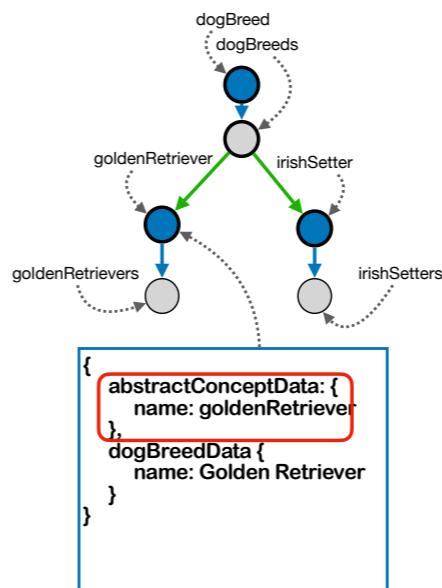
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



abstractConceptData

topological encoding of information INTO the graph

front end

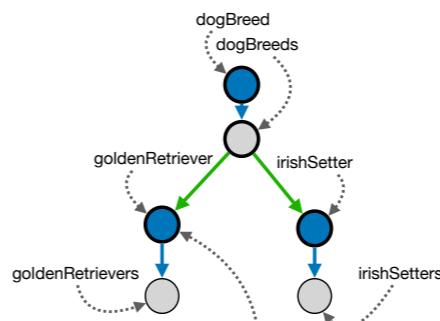
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



```
{ abstractConceptData: { name: goldenRetriever },  
  dogBreedData { name: Golden Retriever } }
```

and dogBreedData.

topological encoding of information INTO the graph

front end

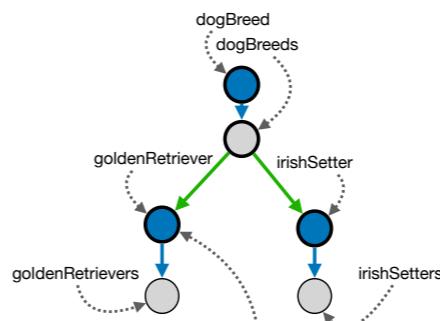
+ concept list

- dogBreed
- goldenRetriever
- irishSetter

+ relationships list

- Irish Setter and Golden Retriever are examples of dog breeds.

back end



```
{  
    abstractConceptData: {  
        name: goldenRetriever  
    },  
    dogBreedData {  
        name: Golden Retriever  
    }  
}
```

As you can see, a single file may serve multiple roles, for example playing the role of representing the abstract idea of one concept, and as being an element of another concept.

[end 16]

17

topological encoding of information INTO the graph relationships between concepts

- one concept forms a subset of another concept
- one concept forms an element of another concept

[start 17]

OK, we have walked through two categories of ways for concepts to relate to one another.

topological encoding of information INTO the graph relationships between concepts

- one concept forms a subset of another concept
- one concept forms an element of another concept
- limit property value to elements of a different concept

The third category that I will show is where the elements of one concept restrict the allowed values of a property of another concept.

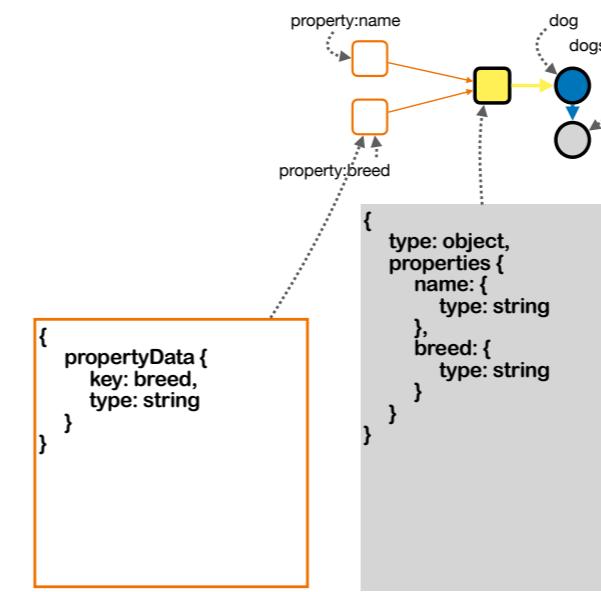
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ breed

back end



Let's go back to the concept of dog, which has properties name and breed, both of which are strings.

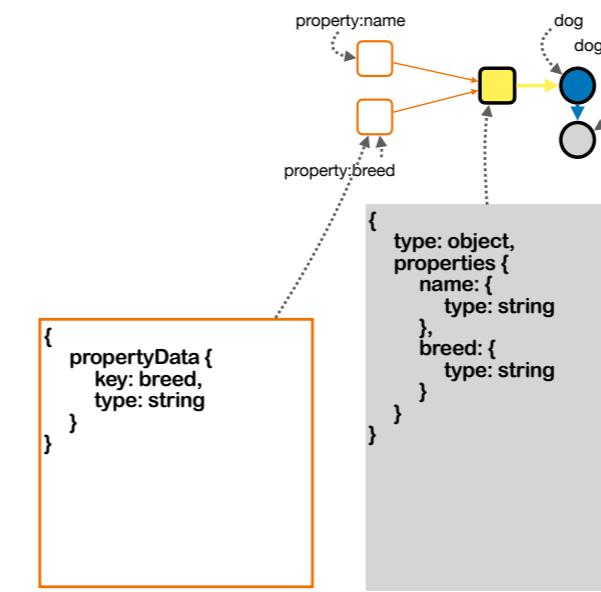
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ breed

back end



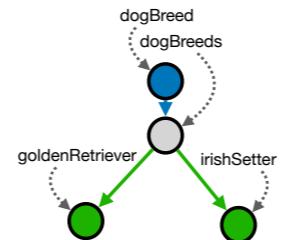
But now the user thinks: I don't want to allow breed to be just any old string.

topological encoding of information INTO the graph

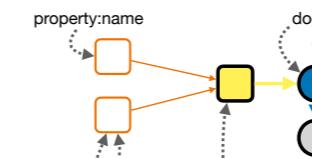
front end

 properties for dog

-  name
-  breed



back end



```
{  
  propertyData {  
    key: breed,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    name: {  
      type: string  
    },  
    breed: {  
      type: string  
    }  
  }  
}
```

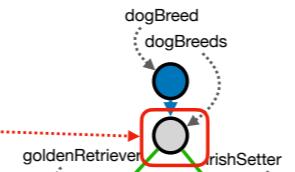
I want it to be limited to breeds in my database of dog breed, like Irish Setter and Golden Retriever, and whatever else I might add.

topological encoding of information INTO the graph

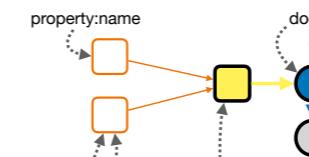
front end

+ properties for dog

name
breed



back end



```
{ propertyData {  
    key: breed,  
    type: string  
}
```

```
{  
    type: object,  
    properties:  
        name: {  
            type: string  
        },  
        breed: {  
            type: string  
        }  
}
```

I don't have any subsets defined, so I will choose the superset of all dog breeds.

topological encoding of information INTO the graph

front end

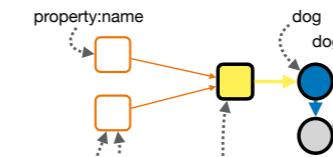
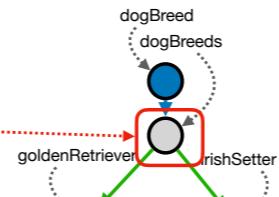
+ properties for dog

name
breed

+ relationships list

- breed is a string

back end



```
{  
  propertyData {  
    key: breed,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    name: {  
      type: string  
    },  
    breed: {  
      type: string  
    }  
  }  
}
```

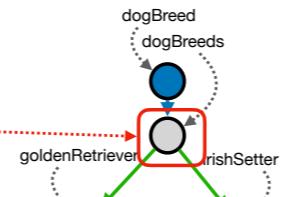
On the front end, the user enters a new relationship. He clicks a few buttons

topological encoding of information INTO the graph

front end

+ properties for dog

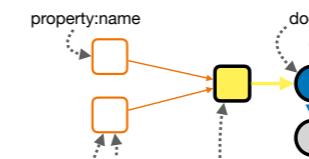
name
breed



+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



```
{ propertyData { key: breed, type: string } }
```

```
{ type: object, properties { name: { type: string }, breed: { type: string } } }
```

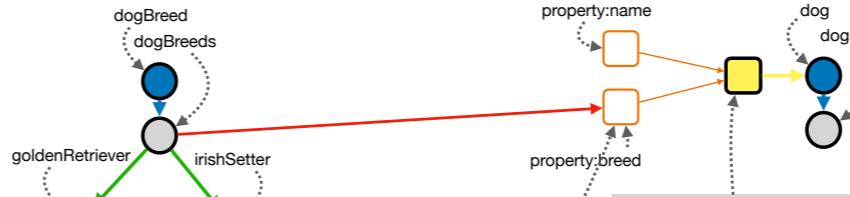
and settles on this human readable relationship, which states that the value of breed is limited to being the name of a single element from the set of dog breeds.

topological encoding of information INTO the graph

front end

properties for dog

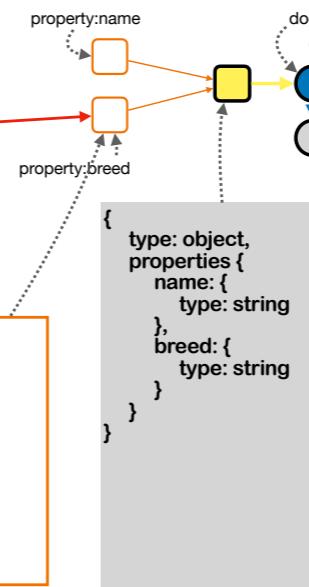
- name
- breed



relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



On the back end, the CG app makes this relationship, represented by the red arrow from the superset node of dogBreed, to the node for property: breed in the concept for dog.

topological encoding of information INTO the graph

front end

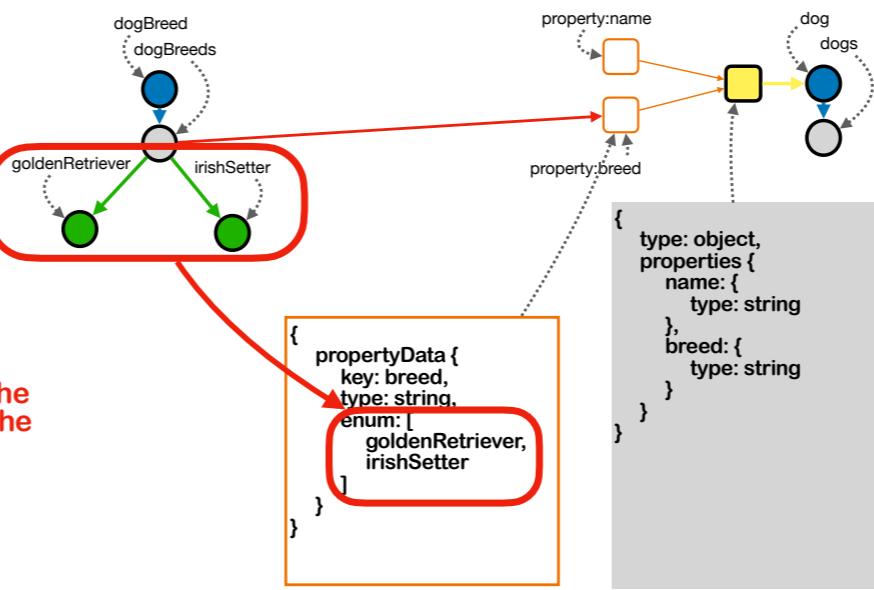
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



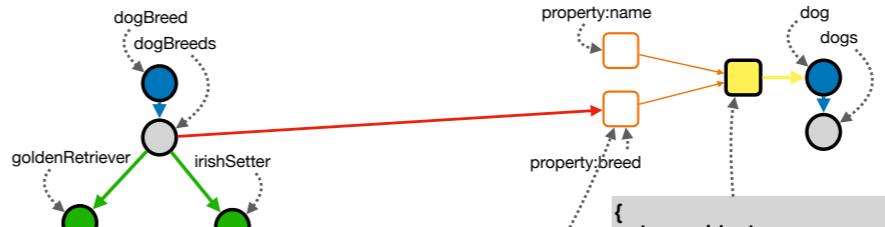
This relationship triggers an update of the propertyData file, where enum limits the value of that property to be whatever is inside enum.

topological encoding of information INTO the graph

front end

+ properties for dog

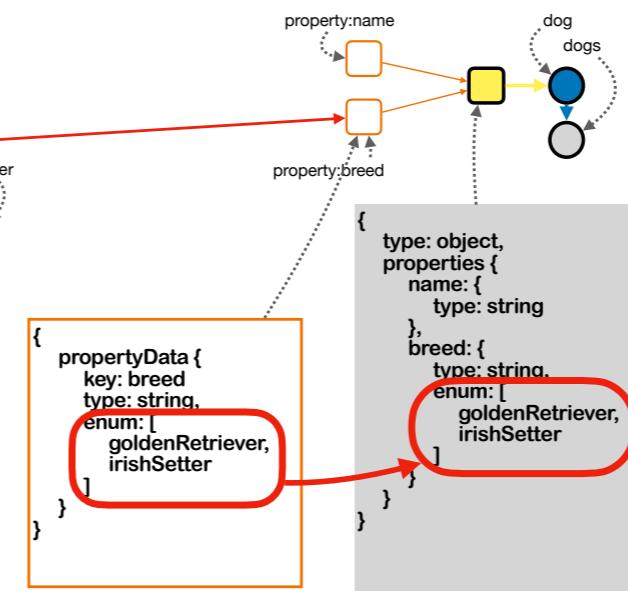
- ❖ name
- ❖ breed



+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



That change propagates to the JSON Schema file.

topological encoding of information INTO the graph

front end

properties for dog

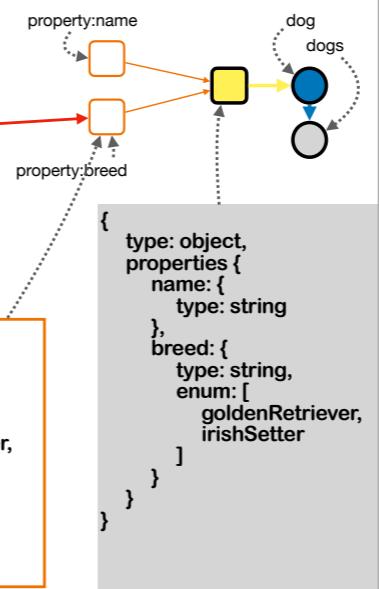
- ❖ name
- ❖ breed



relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



You may be thinking, this graph in the middle looks a little bit familiar.

topological encoding of information INTO the graph

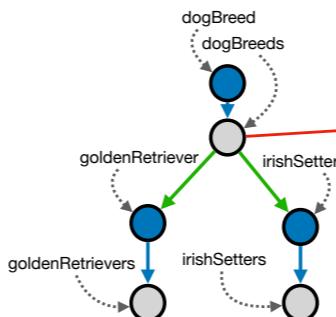
front end

+ properties for dog

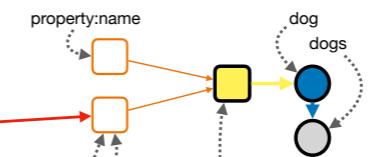
- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{ propertyData {  
    key: breed  
    type: string,  
    enum: [  
        goldenRetriever,  
        irishSetter  
    ]  
}
```

```
{  
    type: object,  
    properties: {  
        name: {  
            type: string  
        },  
        breed: {  
            type: string,  
            enum: [  
                goldenRetriever,  
                irishSetter  
            ]  
        }  
    }  
}
```

A few slides ago, we had previously introduced gR and iS as full-fledged concepts, as shown here.

So let's suppose that we had already done that, and so we have the graph as you see it.

topological encoding of information INTO the graph

front end

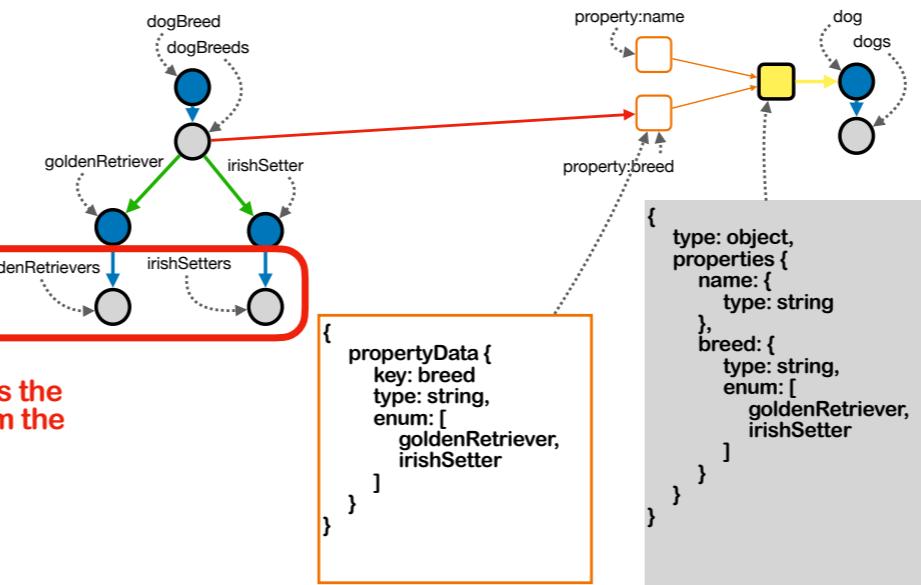
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



In this case, the CG app is able to infer that gR and iS

topological encoding of information INTO the graph

front end

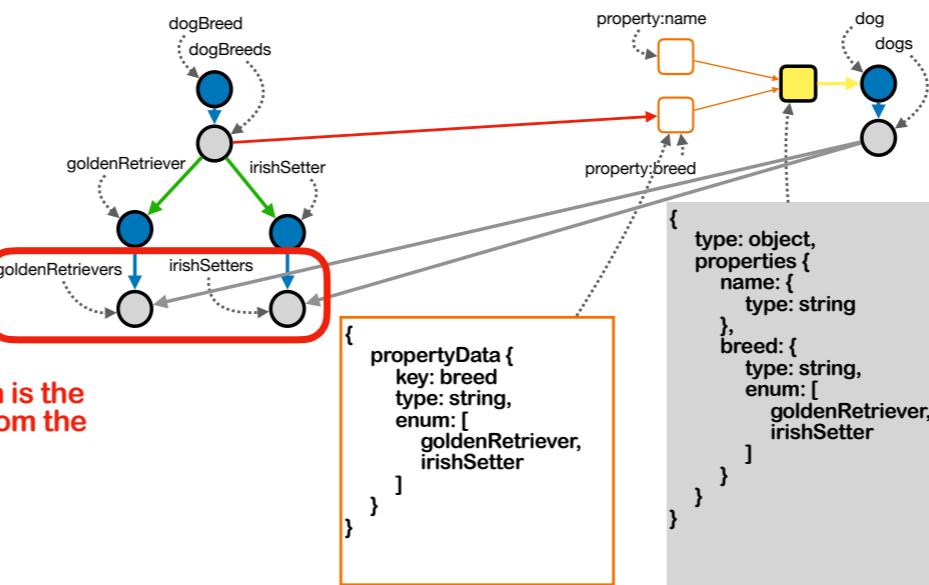
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



are subsets

topological encoding of information INTO the graph

front end

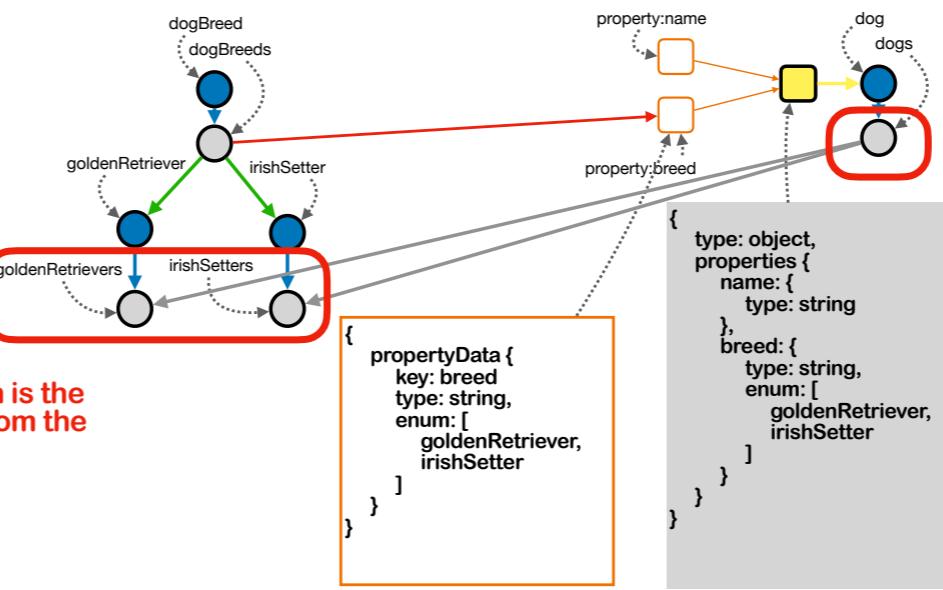
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



of dogs.

topological encoding of information INTO the graph

front end

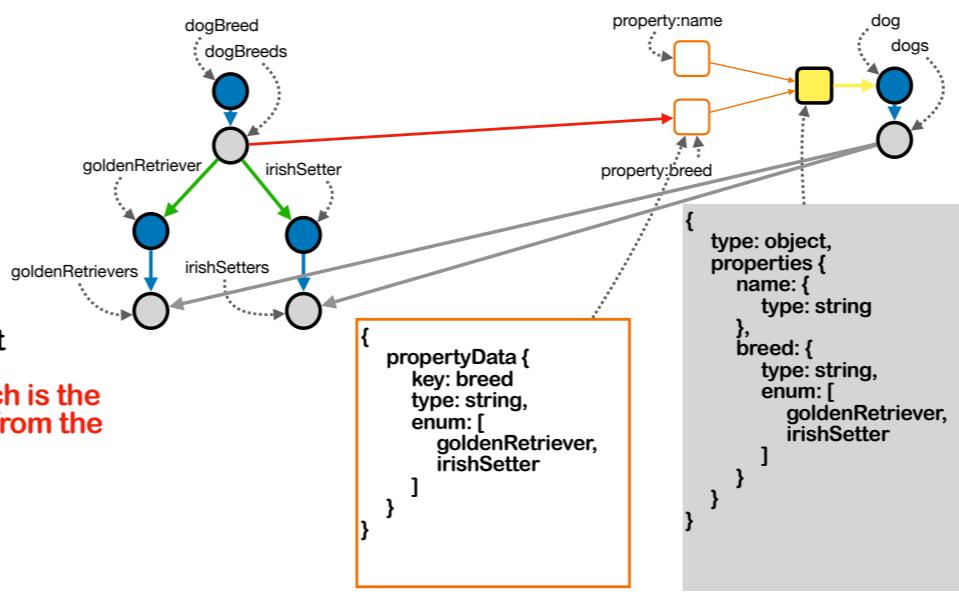
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



It knows this because the configuration of the graph that you see in front of you implies that that must be the case.

topological encoding of information INTO the graph

front end

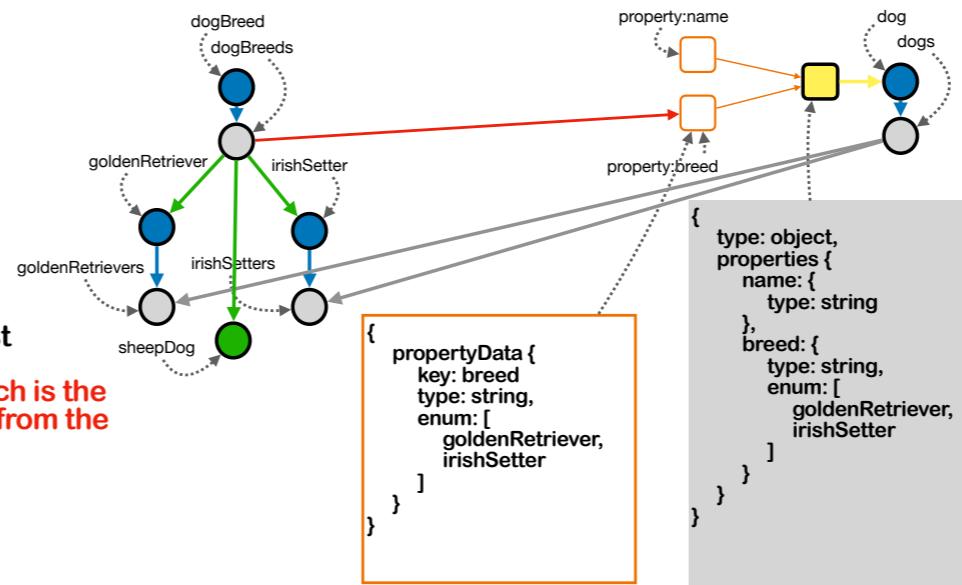
properties for dog

- ❖ name
- ❖ breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



Let's suppose that at some point in the future, the user adds sheep dog to the database of dog breed. The user doesn't recall whether gR and iS had been done as simple elements or as full fledged concepts, but it doesn't matter. The app can infer that sD must be a full fledged concept, bc the app detects that its fellow elements, gR and iS are full fledged concepts.

topological encoding of information INTO the graph

front end

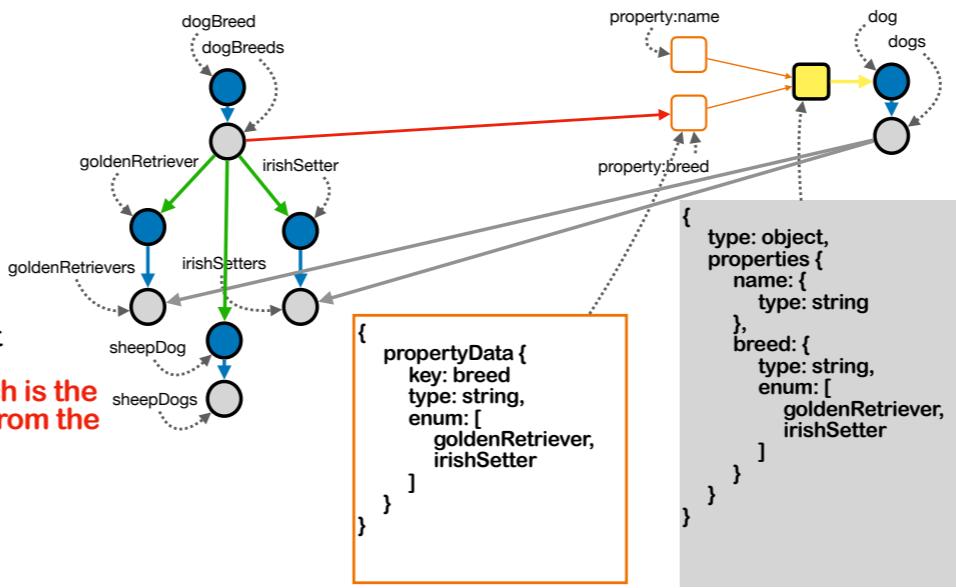
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



So the app, without requiring any additional instruction, automatically adds sheepDogs underneath sheepDog;

topological encoding of information INTO the graph

front end

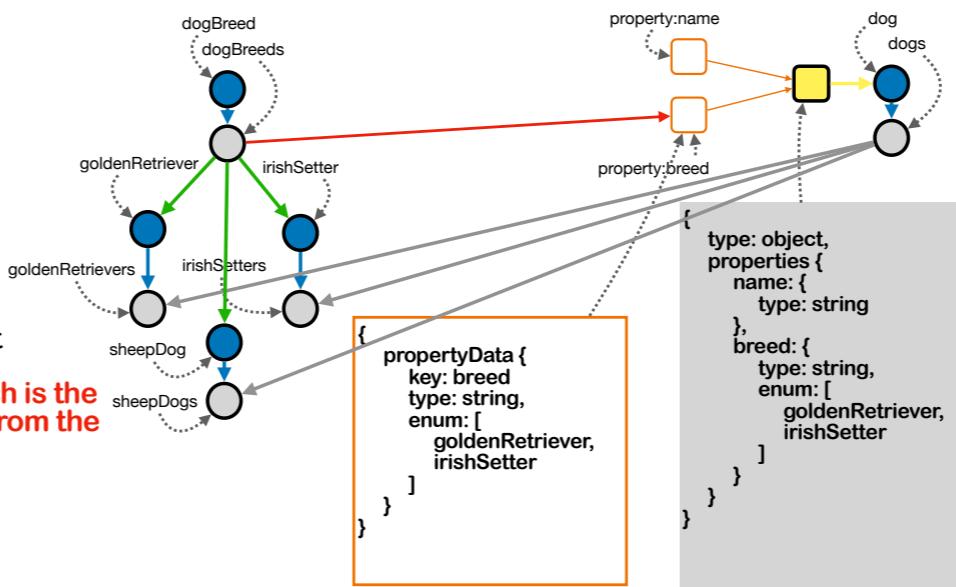
properties for dog

- name
- breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



which in turn triggers the relationship that sD is a subset of dogs.

topological encoding of information INTO the graph

front end

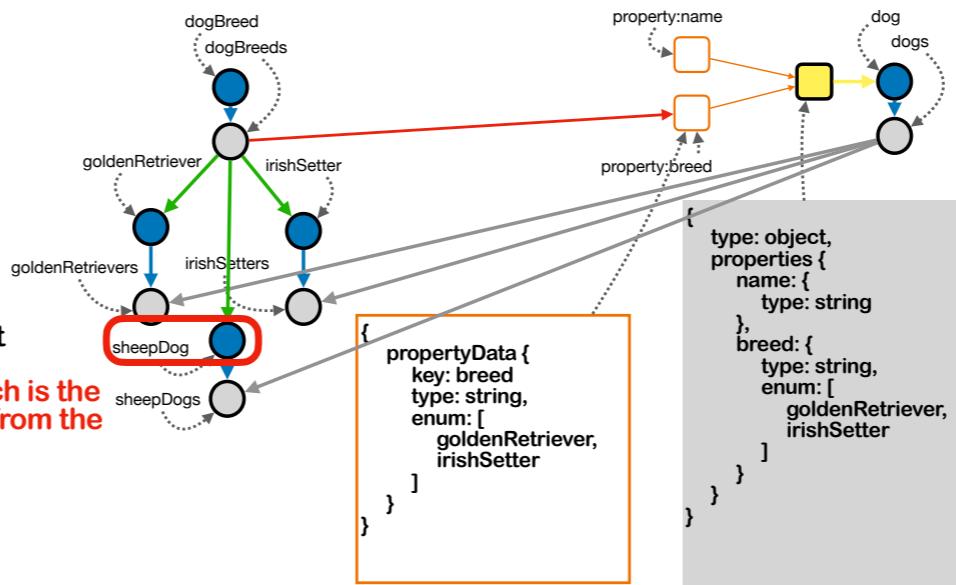
properties for dog

- name
- breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



The app also automatically adds sD

topological encoding of information INTO the graph

front end

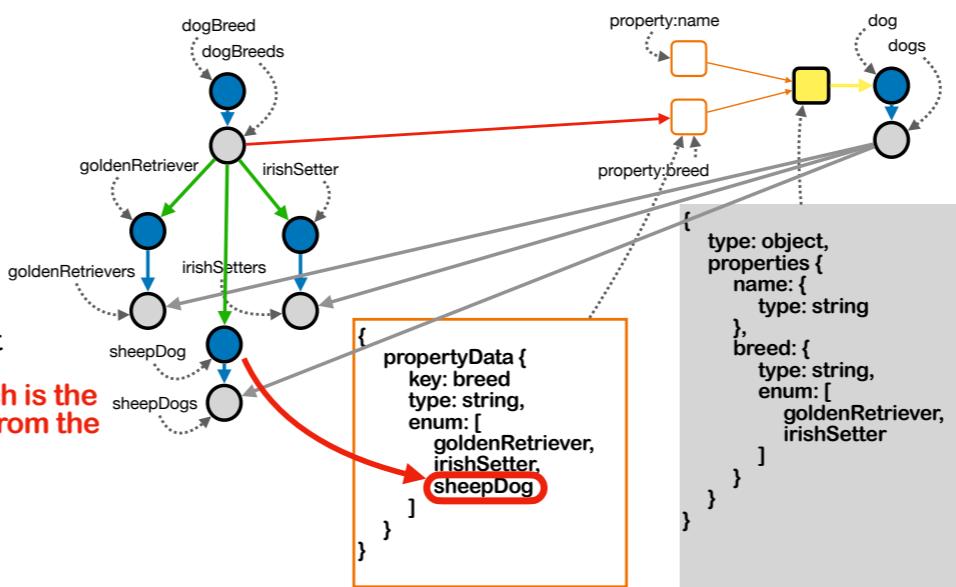
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



to the list inside `propertyData`,

topological encoding of information INTO the graph

front end

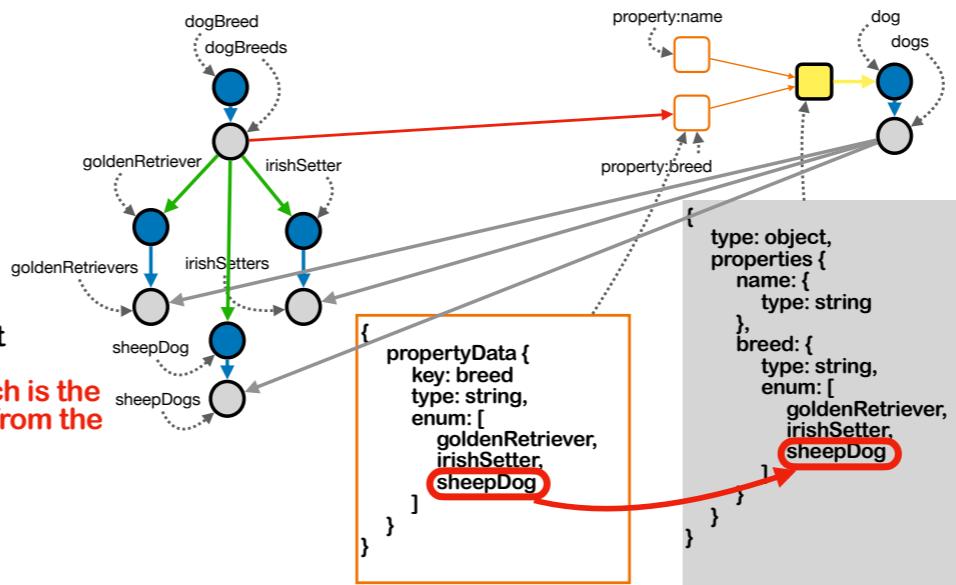
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



and this is automatically propagates to the JSON Schema.

topological encoding of information INTO the graph

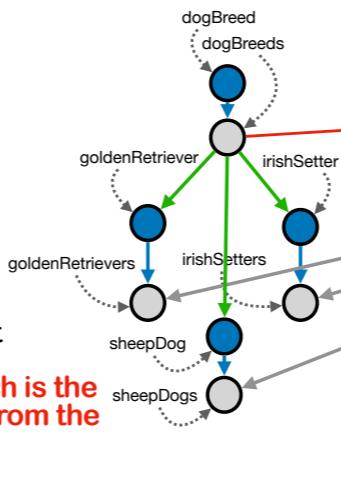
front end

+ properties for dog

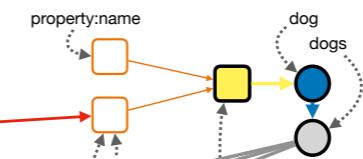
❖ name
❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{ "type": "object", "properties": { "name": { "type": "string" }, "breed": { "type": "string", "enum": [ "goldenRetriever", "irishSetter", "sheepDog" ] } } }
```

I have demonstrated several instances where a change in the graph or a change in one of the JSON files is triggered by some particular configuration in the graph, like a particular type of relationship from a node of some specific type to another node of some particular type.

topological encoding of information INTO the graph

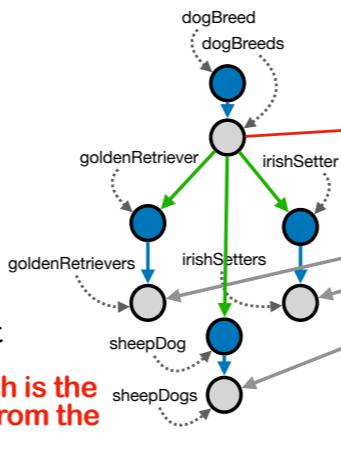
front end

+ properties for dog

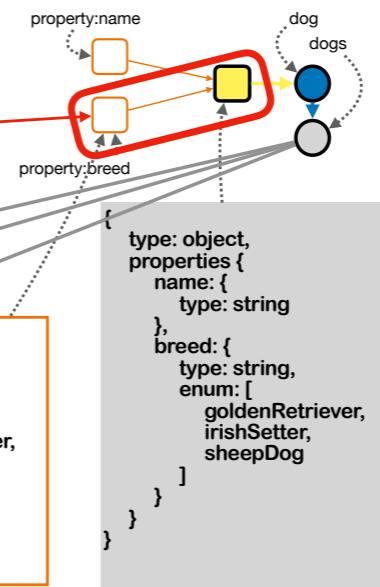
❖ name
❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{ "type": "object", "properties": { "name": { "type": "string" }, "breed": { "type": "string", "enum": [ "goldenRetriever", "irishSetter", "sheepDog" ] } } }
```

As an example, the orange arrow from the node from property: breed to the node for the JSON Schema for dog triggers the CG to check and make sure

topological encoding of information INTO the graph

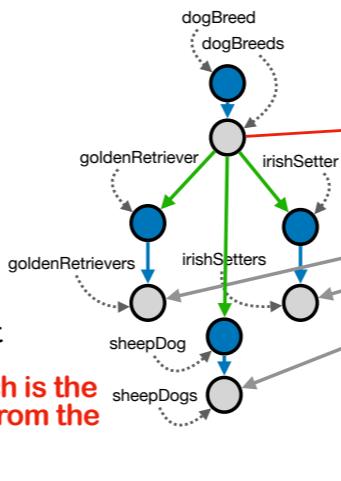
front end

properties for dog

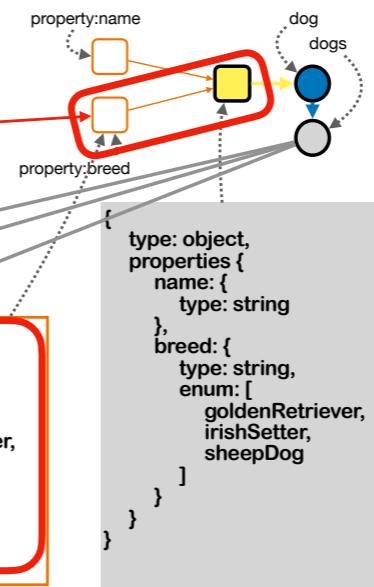
- name
- breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



that the relevant information in the JSON file of the former node

topological encoding of information INTO the graph

front end

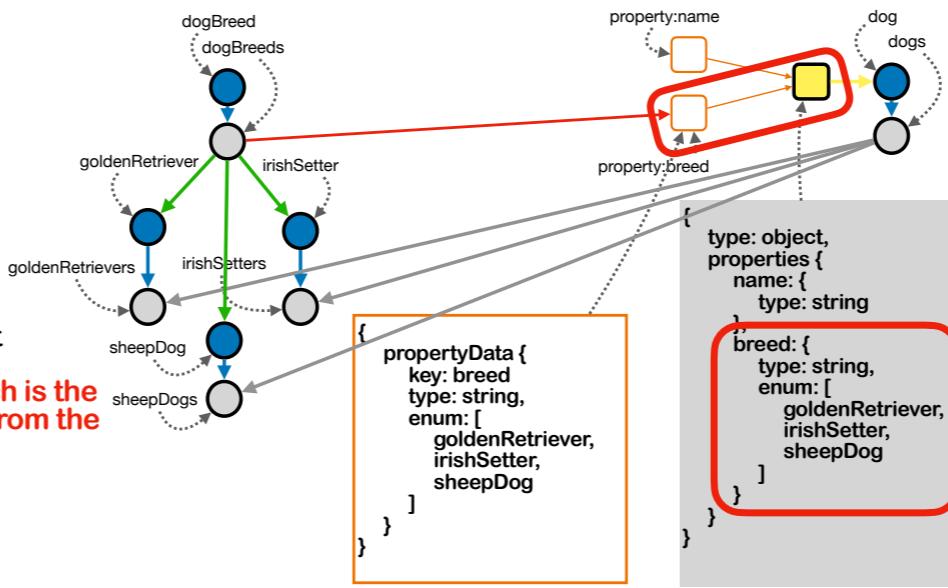
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



has been transferred into the JSON file for the latter node.

topological encoding of information INTO the graph

front end

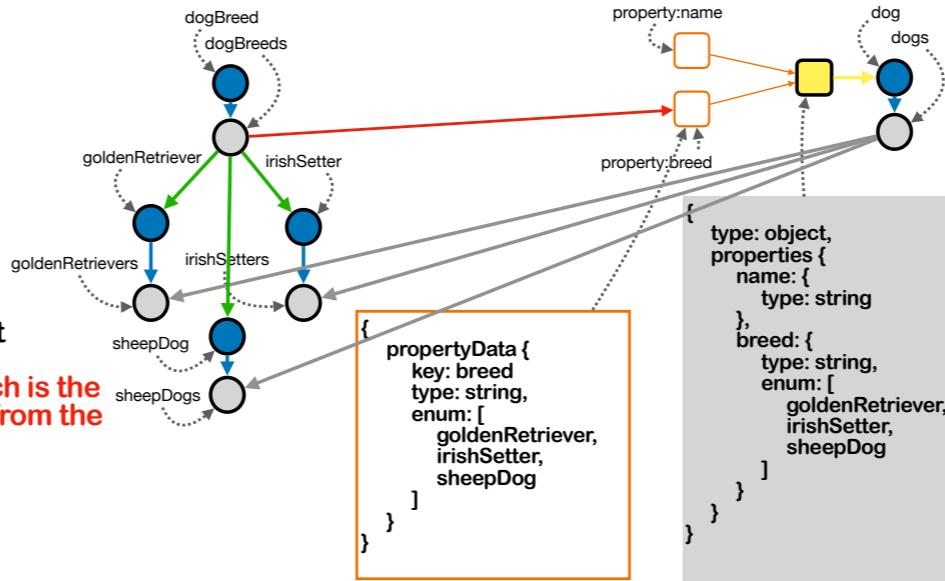
properties for dog

- ❖ name
- ❖ breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



One of the functions of the CG is to periodically cycle through the entire concept graph and make sure that such data transfers have been executed. One thing to know about graph databases is that searching for patterns in a large graph can be computationally expensive. Especially if the pattern you are looking for is a complex one. Therefore it is a design principle that the graphical patterns for which the CG app must search be as simple as possible. Searching for a single link is easier to program, computationally less expensive than searching for an entire Loki Pathway, and less error prone overall. So wherever possible, that is what we do: minimize as much as we can, searches for complex topological patterns; replace it with searches for simple patterns, if possible. I'm not going to go through the details now, but it is possible when updating JSON files as I have shown in a few examples, to avoid routine searches for entire Loki Pathways, and replace that with computationally less expensive searches for just one link at a time within Loki Pathways. Elements crawl their way up the Loki Pathway, one node at a time; that's basically how it works.

topological encoding of information INTO the graph

front end

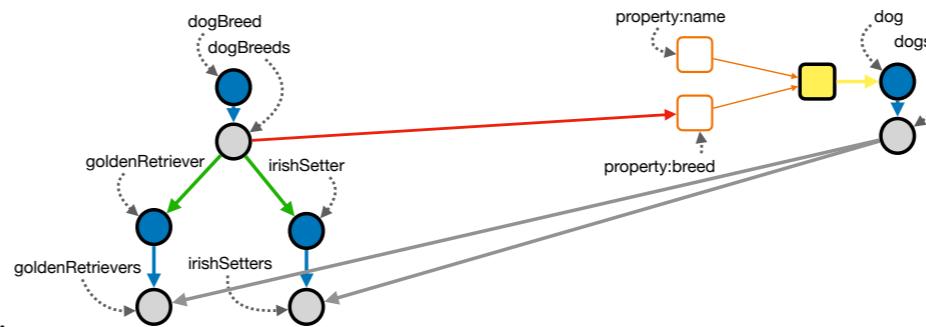
properties for dog

- ❖ name
- ❖ breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



There is another implication of adding subset relationships.

topological encoding of information INTO the graph

front end

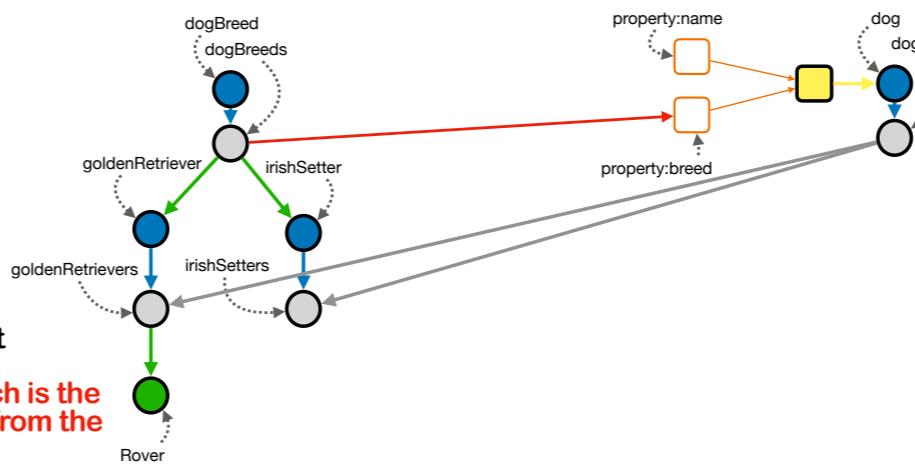
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



Suppose the user had previously entered Rover into the database under golden retriever, but did not enter that Rover was a dog or that Rover was an animal because, for example, imagine that those concepts had not yet been created.

topological encoding of information INTO the graph

front end

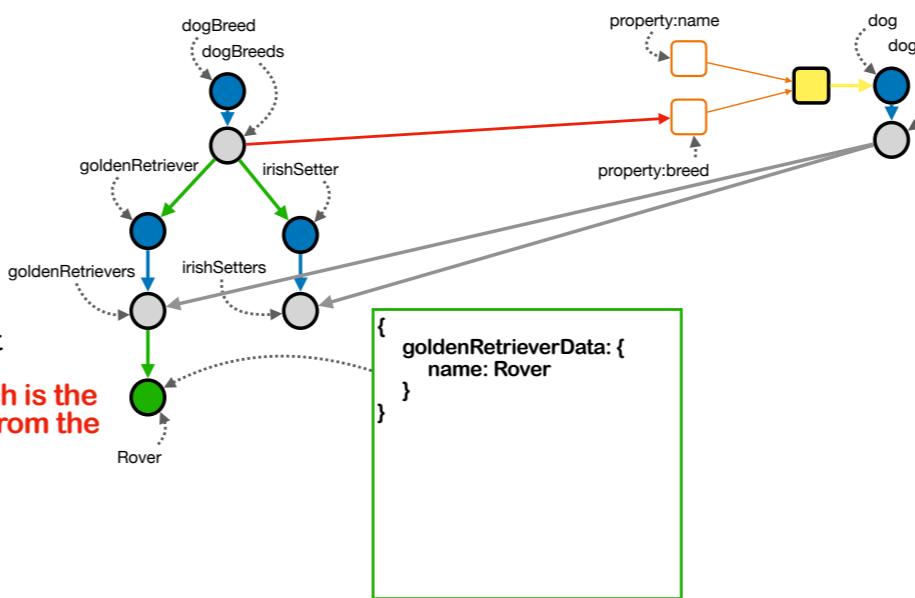
properties for dog

- ❖ name
- ❖ breed

relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



The file for Rover would look like this, with a property with the key: goldenRetrieverData.

topological encoding of information INTO the graph

front end

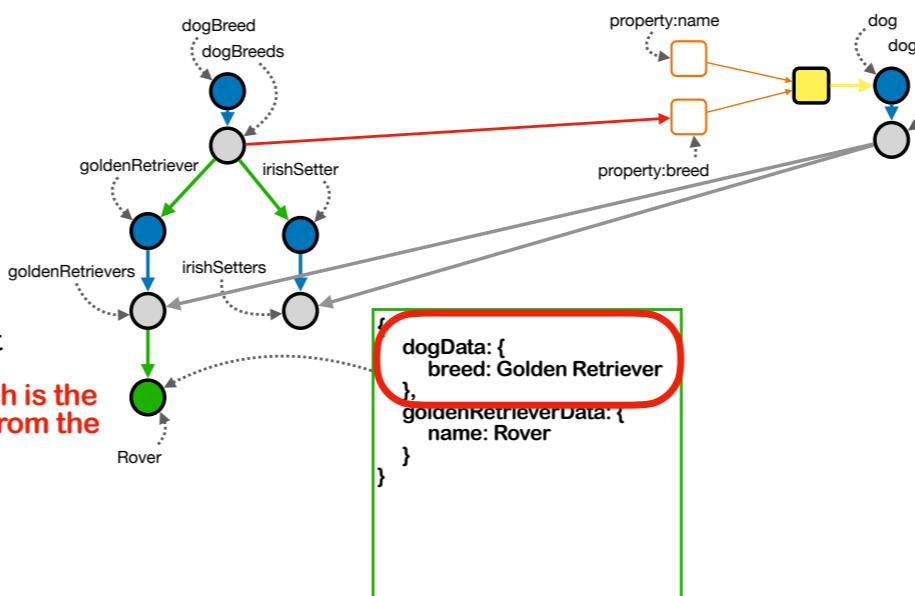
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



the CG, by looking at the graph, now knows that Rover, not only is a GR, but that Rover is a dog. Therefore the file for Rover must also have a property for dogData. And the app knows automatically that within dogData property, there is a key: breed, and its value must be Golden Retriever.

topological encoding of information INTO the graph

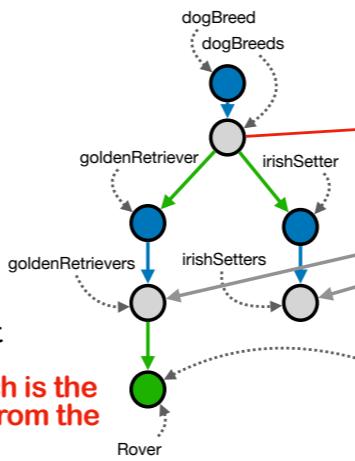
front end

+ properties for dog

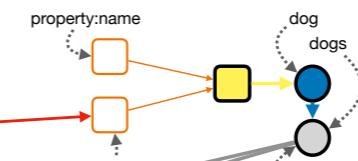
- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{  
  dogData: {  
    breed: Golden Retriever  
  },  
  goldenRetrieverData: {  
    name: Rover  
  }  
}
```

```
{  
  supersetData: {  
    name: dogs,  
    elements: [  
    ]  
  }  
}
```

the file for the superset of all dogs also must be updated

topological encoding of information INTO the graph

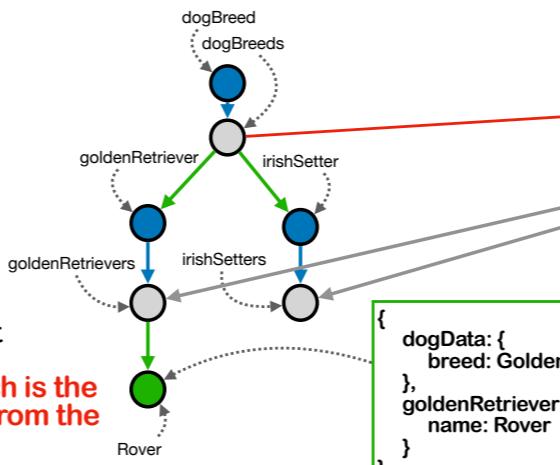
front end

+ properties for dog

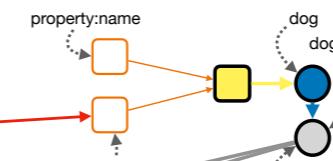
- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{  
  dogData: {  
    breed: Golden Retriever  
  },  
  goldenRetrieverData: {  
    name: Rover  
  }  
}
```

```
{  
  supersetData: {  
    name: dogs,  
    elements: [  
      Rover  
    ]  
  }  
}
```

by adding Rover as an element.

topological encoding of information INTO the graph

front end

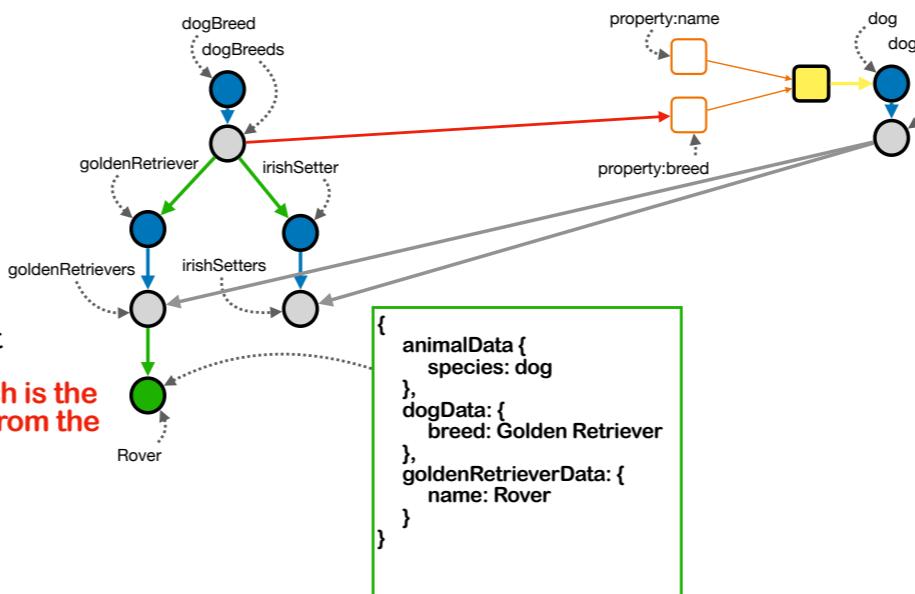
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



The CG also updates that Rover is an animal,

topological encoding of information INTO the graph

front end

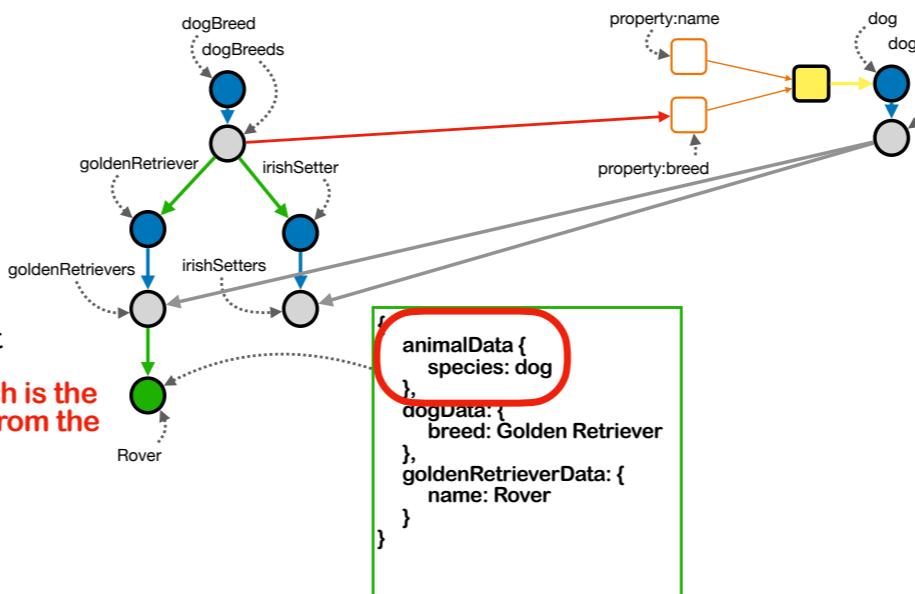
+ properties for dog

- ❖ name
- ❖ breed

+ relationships list

- breed is a string which is the name of an element from the set of dog breeds.

back end



by modifying the file for Rover,

topological encoding of information INTO the graph

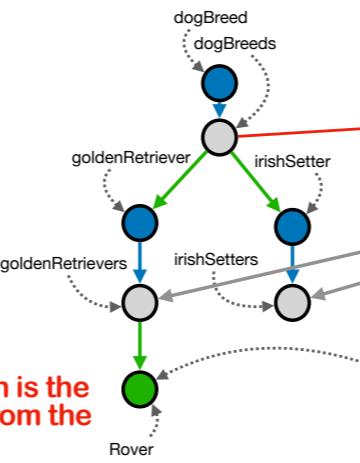
front end

[+] properties for dog

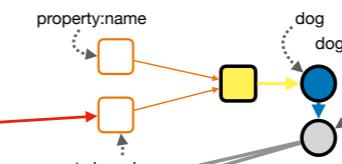
- ❖ name
- ❖ breed

[+] relationships list

- breed is a string which is the name of an element from the set of dog breeds.



back end



```
{  
    animalData {  
        species: dog  
    },  
    dogData: {  
        breed: Golden Retriever  
    },  
    goldenRetrieverData: {  
        name: Rover  
    }  
}
```

and also modifying the file for the superset of all animals (which I have not shown here).

[end 17]

18

topological encoding of information INTO the graph

front end

back end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

[start 18]

So now let's imagine we have gone through this process of setting up multiple properties for the concept for dog. The first one, name, we keep as a string; but the rest of them we have chosen to restrict to being elements of other concepts. What does this look like in the back end?

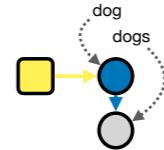
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



On the back end, we have here the JSON Schema for dog,

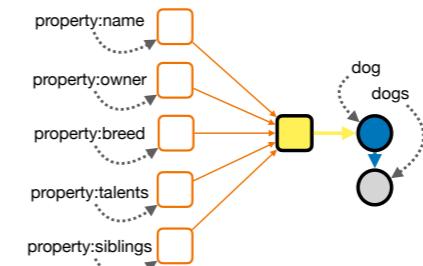
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



and it takes input from five property nodes.

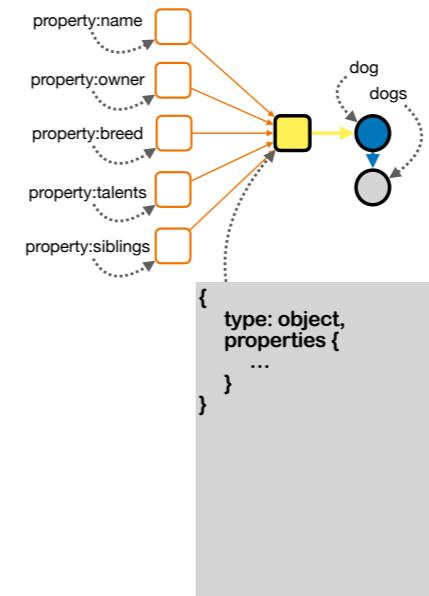
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



Let's look at those 5 inputs, one at a time.

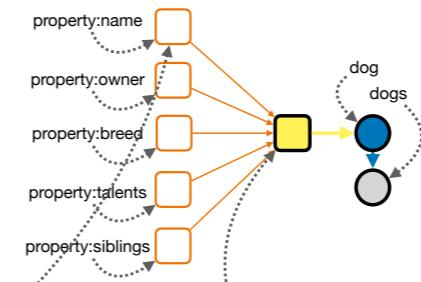
topological encoding of information INTO the graph

front end

 properties for dog

-  name
-  owner
-  breed
-  talents
-  siblings

back end



```
{  
  propertyData {  
    key: name,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    ...  
  }  
}
```

First we have name, which is a simple string.

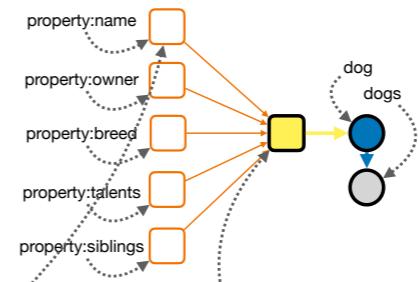
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: name,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties:  
    name: {  
      type: string  
    }  
  }  
}
```

It is inserted into the JSON Schema like this.

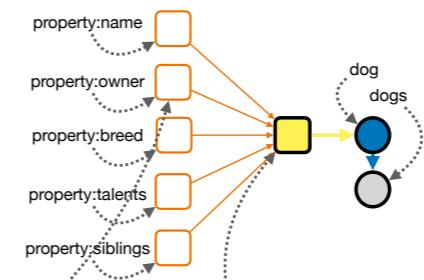
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    ...  
  }  
}
```

Next we see owner, also a simple string.

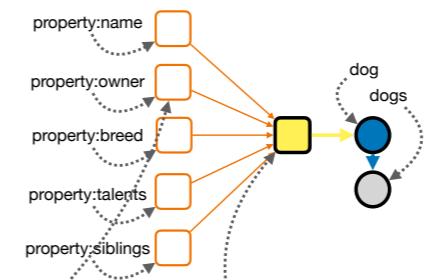
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties:  
    owner: {  
      type: string  
    }  
  }  
}
```

And it is inserted into JSON Schema like this.

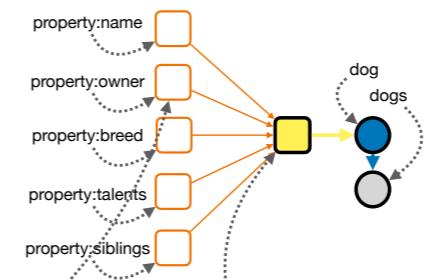
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    owner: {  
      type: string  
    }  
  }  
}
```

(Note that I am showing only one property at a time in the JSON Schema,

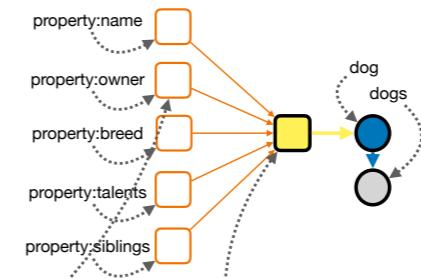
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    name: {  
      type: string  
    },  
    owner: {  
      type: string  
    },  
    breed: {  
      type: string  
    },  
    talents: {  
      type: array  
    }  
  }  
}
```

even though ultimately all 5 will be present. As you can see, they don't all fit on the slide.)

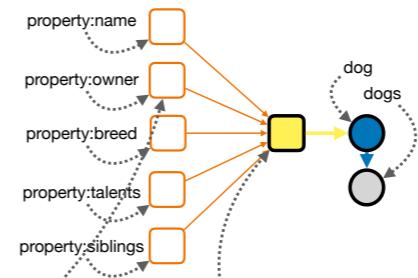
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    owner: {  
      type: string  
    }  
  }  
}
```

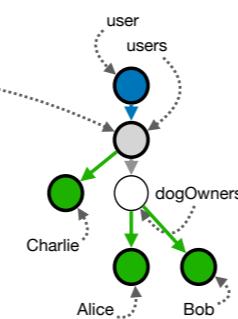
The user thinks: I want to restrict owner to be an element of the set of users.

topological encoding of information INTO the graph

front end

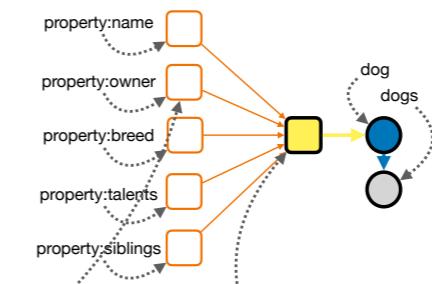
properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

back end



```
{  
  type: object,  
  properties {  
    owner: {  
      type: string  
    }  
  }  
}
```

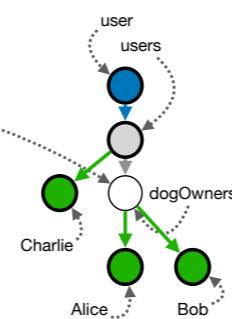
Like this.

topological encoding of information INTO the graph

front end

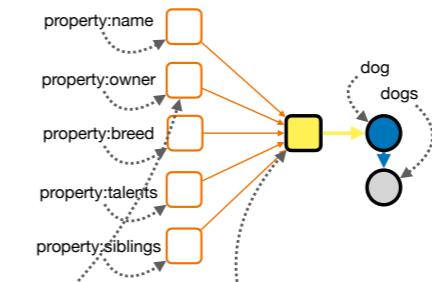
properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



```
{  
  propertyData {  
    key: owner,  
    type: string  
  }  
}
```

back end



```
{  
  type: object,  
  properties {  
    owner: {  
      type: string  
    }  
  }  
}
```

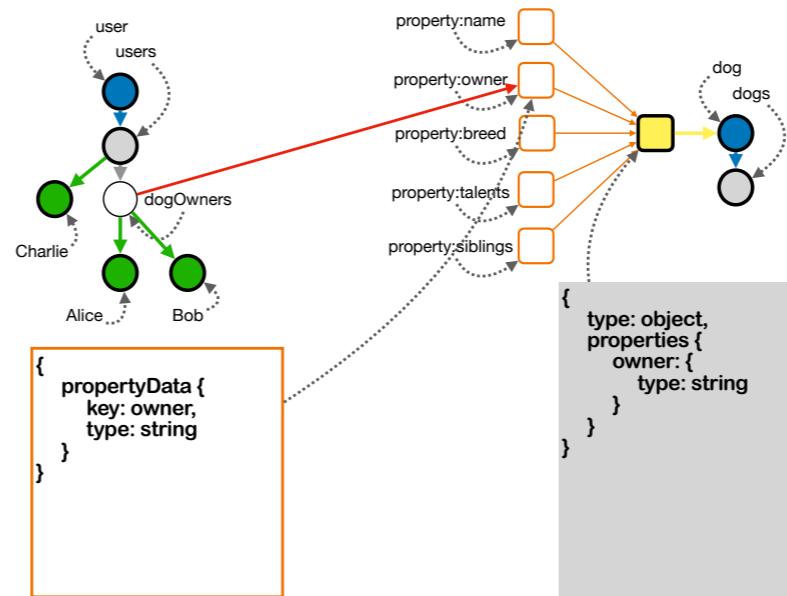
But hey, I've got a subset of users labelled dogOwners, so I'll restrict to that subset instead.

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



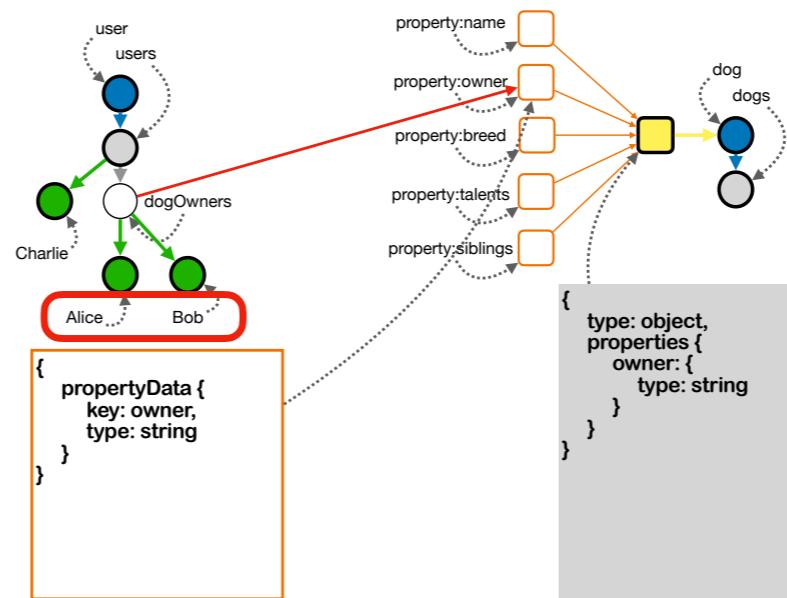
In the back end, the CG makes this relationship, shown with the red arrow, from the node for dogOwners to the orange node representing the property: owner.

topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

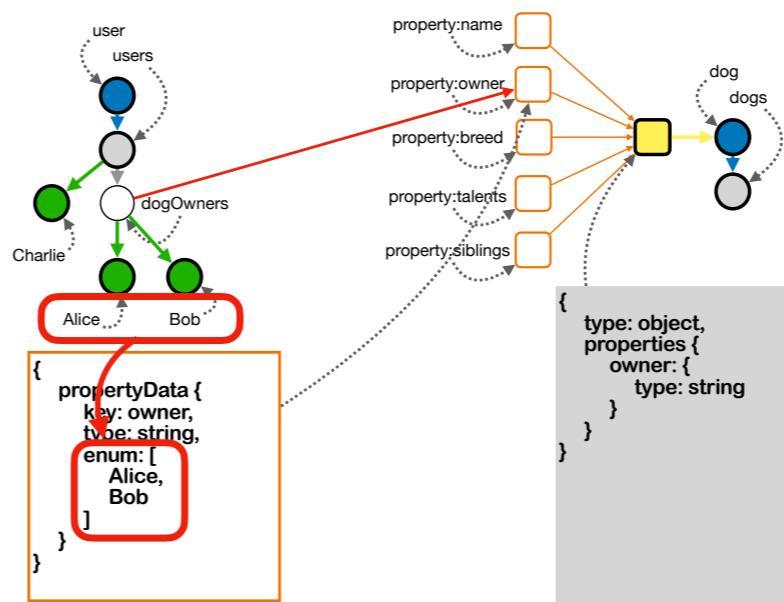
This triggers the elements of the set of dogOwners, Alice and Bob,

topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



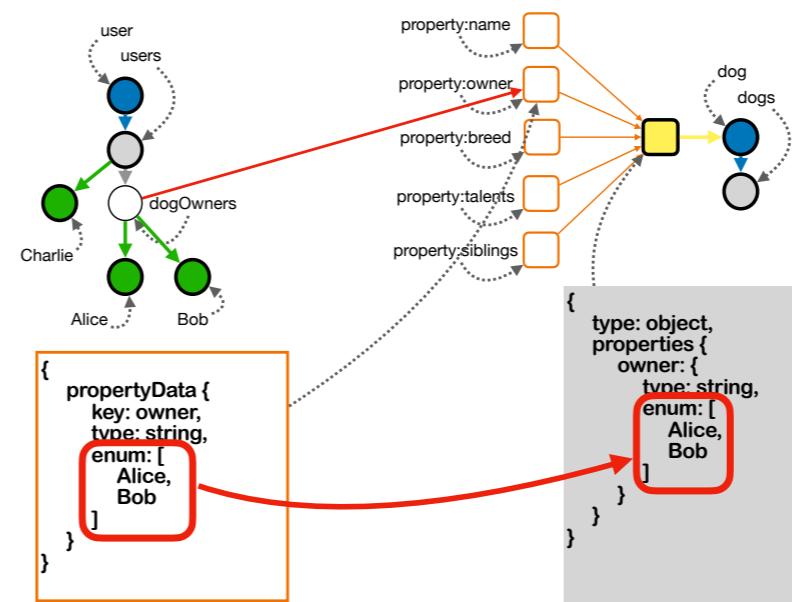
to be inserted into the proper place under propertyData,

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

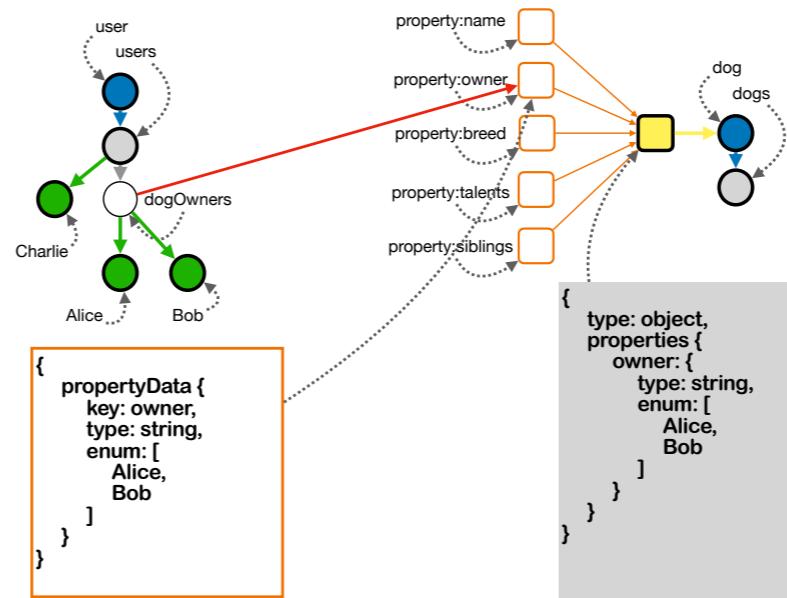
which then propagates to the proper place in the JSON Schema.

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

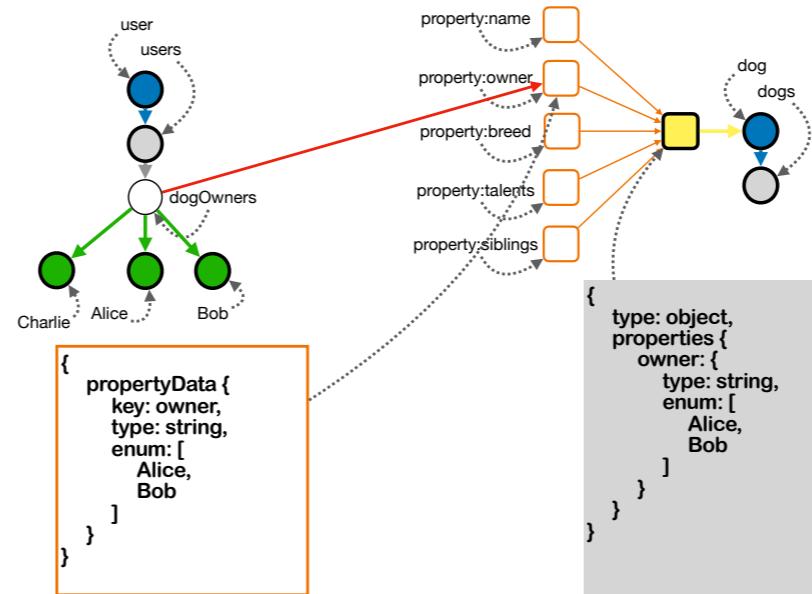
Suppose at some later point in time, Charlie gets moved ...

topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



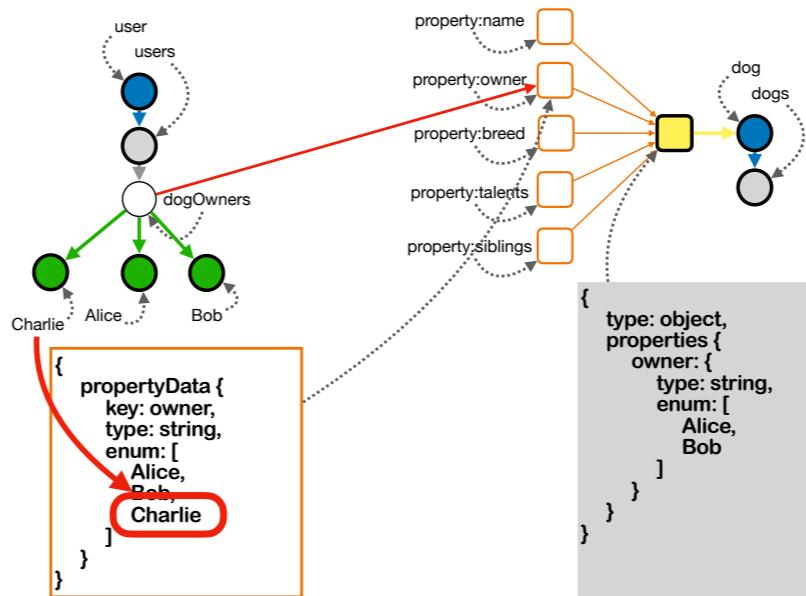
into the set of dogOwners.

topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

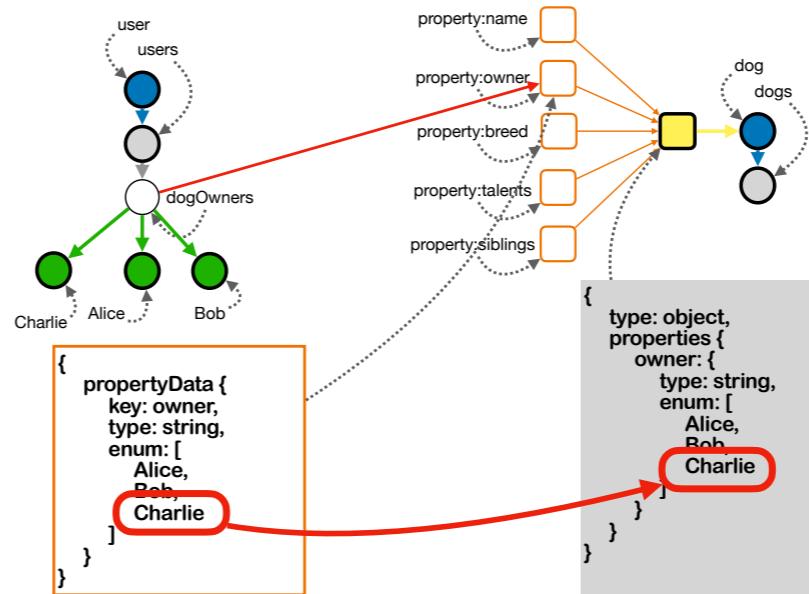
This triggers Charlie to get inserted into the list inside propertyData,

topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

which triggers the automatic propagation of this element into the corresponding list in the JSON Schema.

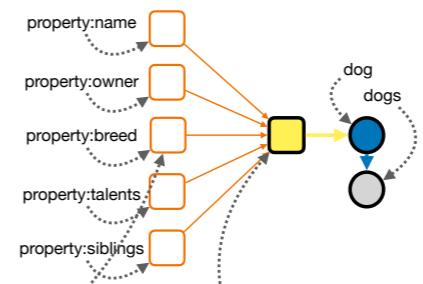
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ **breed**
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: breed,  
    type: string  
  }  
}
```

```
{  
  type: object,  
  properties {  
    breed: {  
      type: string  
    }  
  }  
}
```

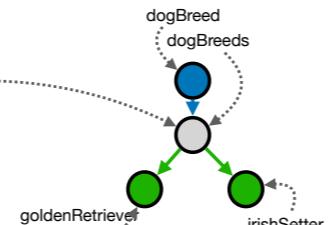
We will go through the same process with breed.

topological encoding of information INTO the graph

front end

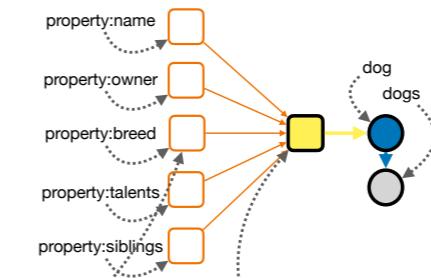
+ properties for dog

- ❖ name
- ❖ owner
- ❖ **breed**
- ❖ talents
- ❖ siblings



```
{  
  propertyData {  
    key: breed,  
    type: string  
  }  
}
```

back end



```
{  
  type: object,  
  properties {  
    breed: {  
      type: string  
    }  
  }  
}
```

The user decides to restrict the value for breed to be elements of the concept for dogBreed.

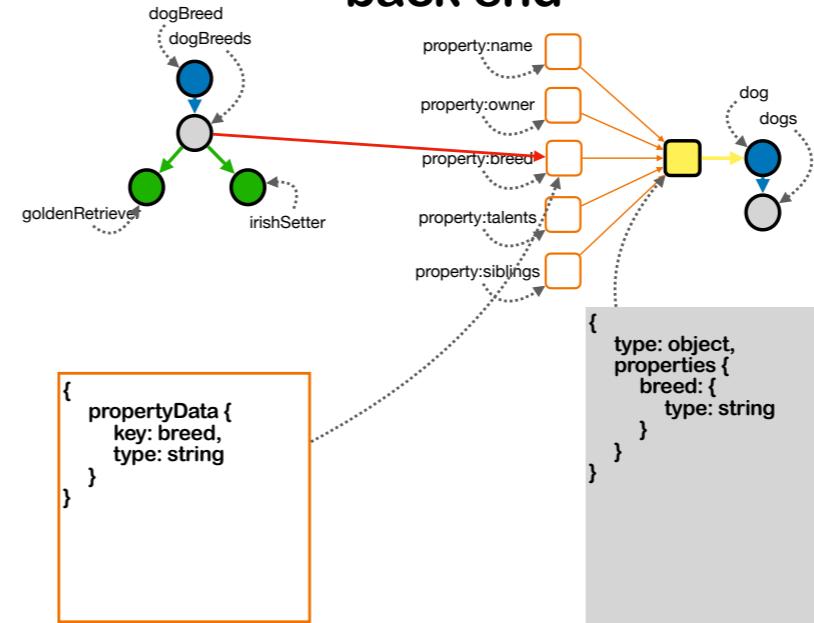
topological encoding of information INTO the graph

front end

 properties for dog

-  name
-  owner
-  breed
-  talents
-  siblings

back end



At the back end, this triggers this relationship to be created pointing from the node for dogBreeds to the node for the property: breed.

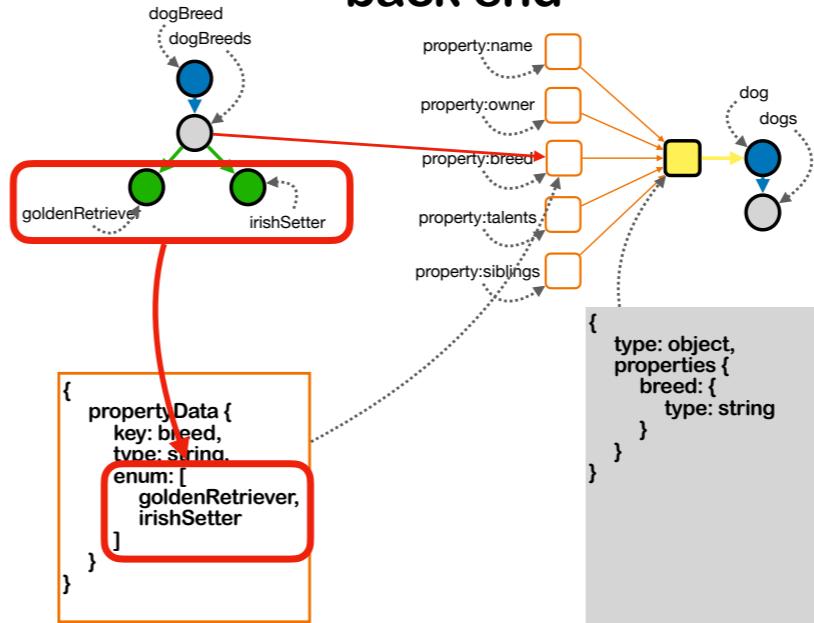
topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ **breed**
- ❖ talents
- ❖ siblings

back end



This triggers the elements of dogBreeds to be added into the JSON file for the property for breed.

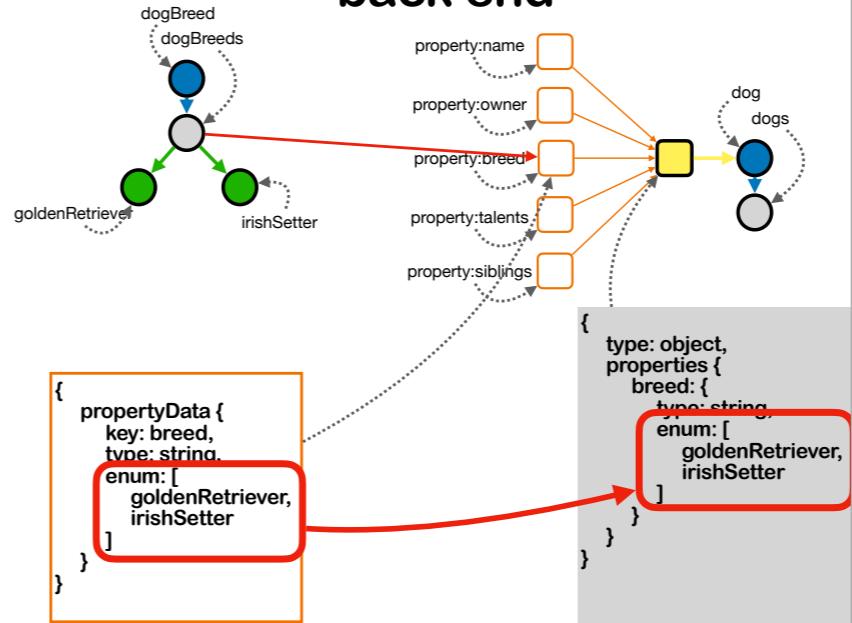
topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ **breed**
- ❖ talents
- ❖ siblings

back end



and these elements propagate automatically into the JSON Schema.

[end 18]

19

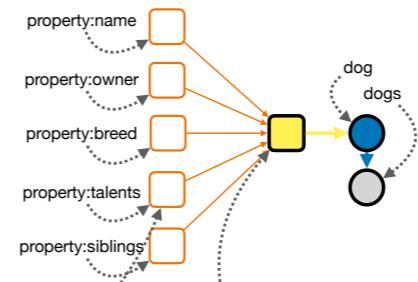
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: talents,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    talents: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

[start 19]

Something similar is going to happen with the property for talents, but it is different in one important way. The property for talents is not single valued, like name and owner, but multi valued, bc a dog can have anywhere from 0 to a great number of talents.

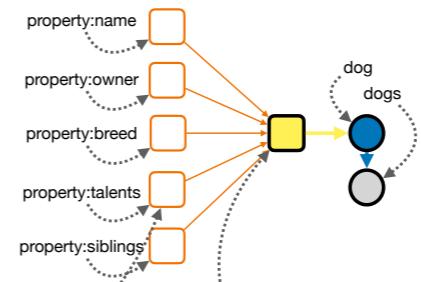
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: talents  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    talents: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

So instead of talents being a string, it is an array, and the items of the array are strings.

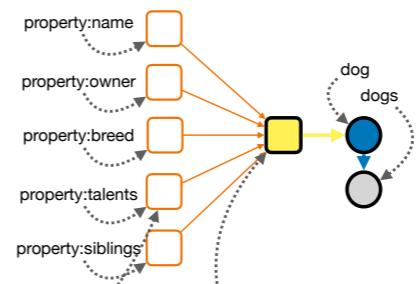
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: talents,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    talents: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

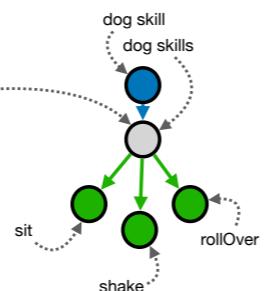
The user decides that these items, instead of being allowed to be any old strings,

topological encoding of information INTO the graph

front end

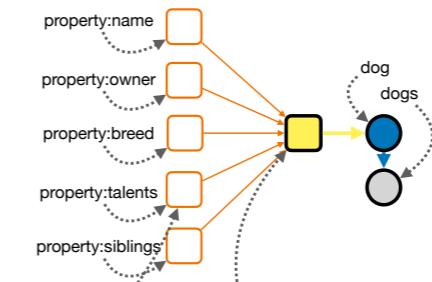
properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



```
{  
  propertyData {  
    key: talents,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

back end



```
{  
  type: object,  
  properties {  
    talents: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

should be restricted to elements of the set of dog skills.

topological encoding of information INTO the graph

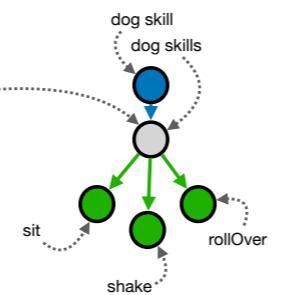
front end

+ properties for dog

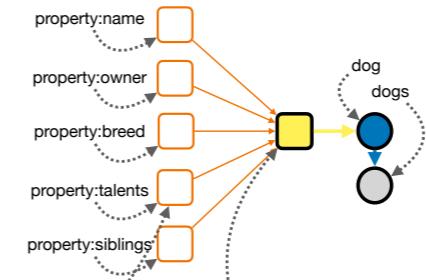
- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

+ relationships list

- Items of talents are strings



back end



```
{  
  "propertyData": {  
    "key": "talents",  
    "type": "array",  
    "items": {  
      "type": "string"  
    }  
  }  
}
```

```
{  
  "type": "object",  
  "properties": {  
    "talents": {  
      "type": "array",  
      "items": {  
        "type": "string"  
      }  
    }  
  }  
}
```

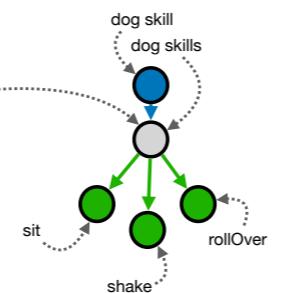
The user clicks some buttons

topological encoding of information INTO the graph

front end

[+] properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

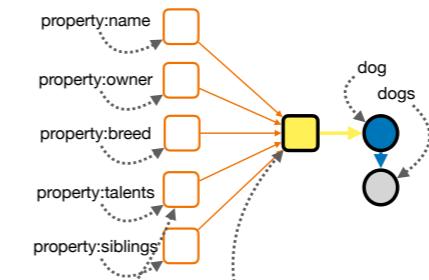


[+] relationships list

- Items of talents are strings which are restricted to be the name of an element from the set of dog skills.

```
{  
  propertyData {  
    key: talents,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

back end



```
{  
  type: object,  
  properties {  
    talents: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

and creates a new relationship. The user sees this as a sentence, shown here in red, which can be understood intuitively. Note that this sentence is worded a little differently than the last few relationships, since talents is an ARRAY of strings, not a single string like owner and breed.

topological encoding of information INTO the graph

front end

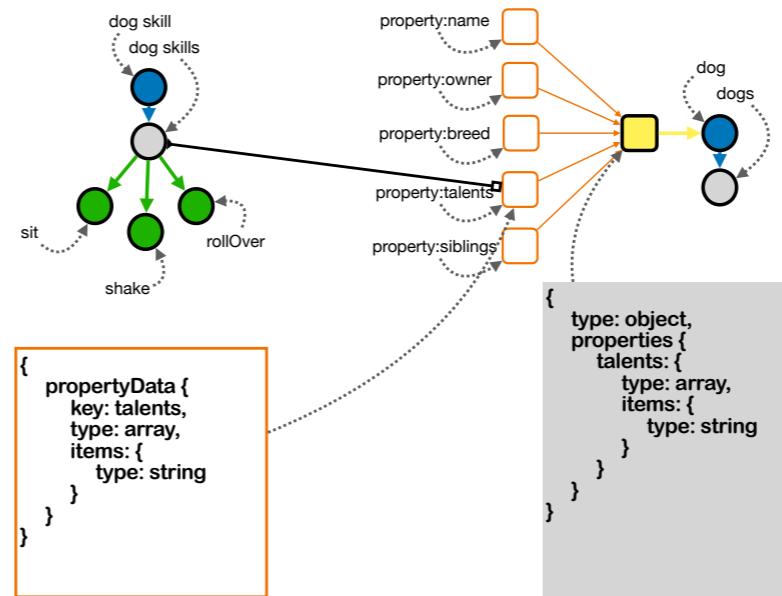
[+] properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

[+] relationships list

- Items of talents are strings which are restricted to be the name of an element from the set of dog skills.

back end



And the subsequent relationship which is triggered at the back end is also a little different. Similar to previous steps, the relationship goes from the grey node for dogSkills to the orange node for the property for talents.

topological encoding of information INTO the graph

front end

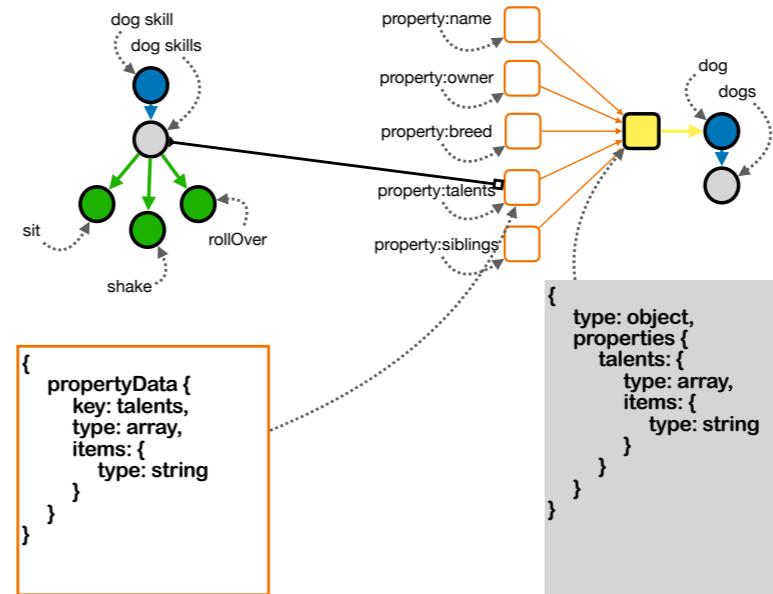
[+] properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

[+] relationships list

- Items of talents are strings which are restricted to be the name of an element from the set of dog skills.

back end



But the relationship is a different one, as indicated by the different color (a black arrow instead of a red arrow). The CG knows that wherever it sees this relationship, the property to which it points

topological encoding of information INTO the graph

front end

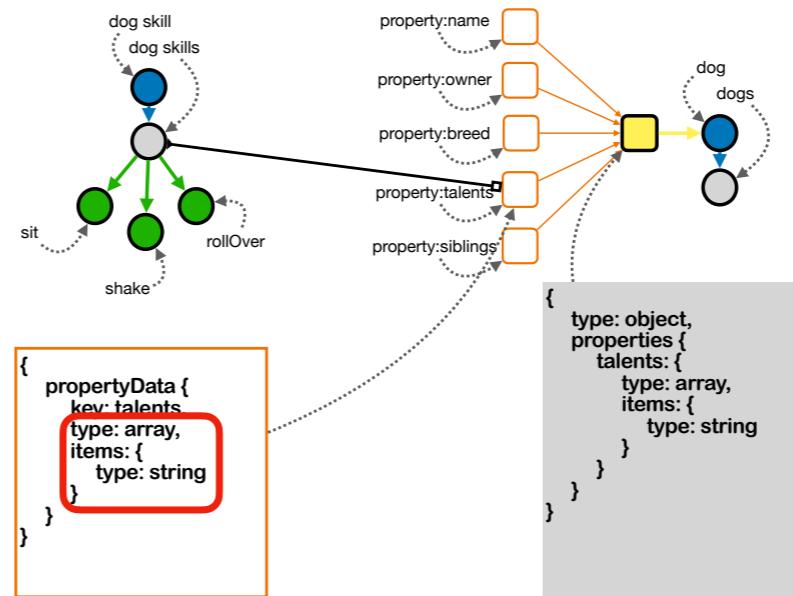
[+] properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

[+] relationships list

- Items of talents are strings which are restricted to be the name of an element from the set of dog skills.

back end



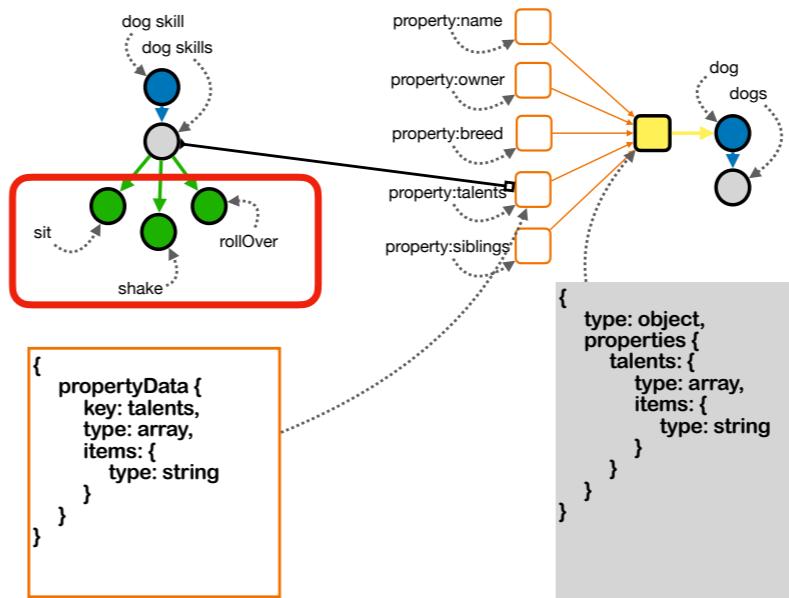
is of type:array, not string. (The property node for talents should already be an array, but if it were not already so, the CG would make the correction.)

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



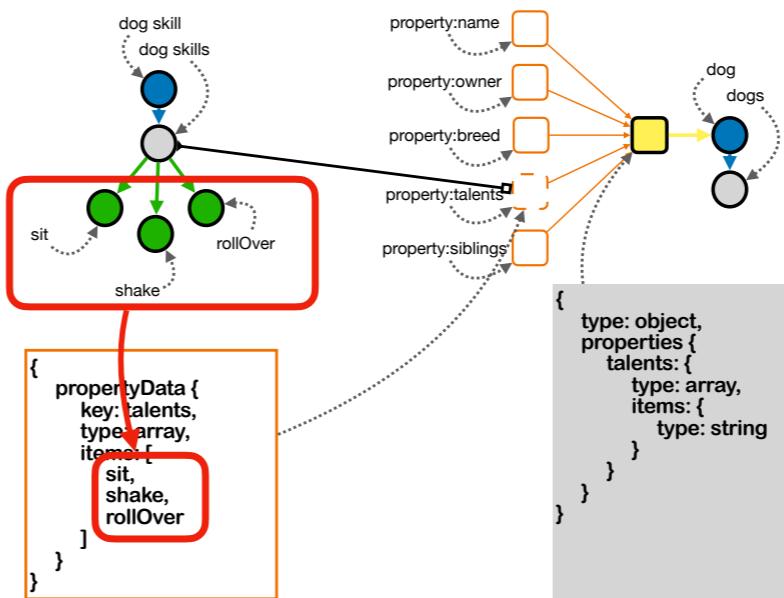
The black arrow relationship triggers the elements of dogSkills

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



back end

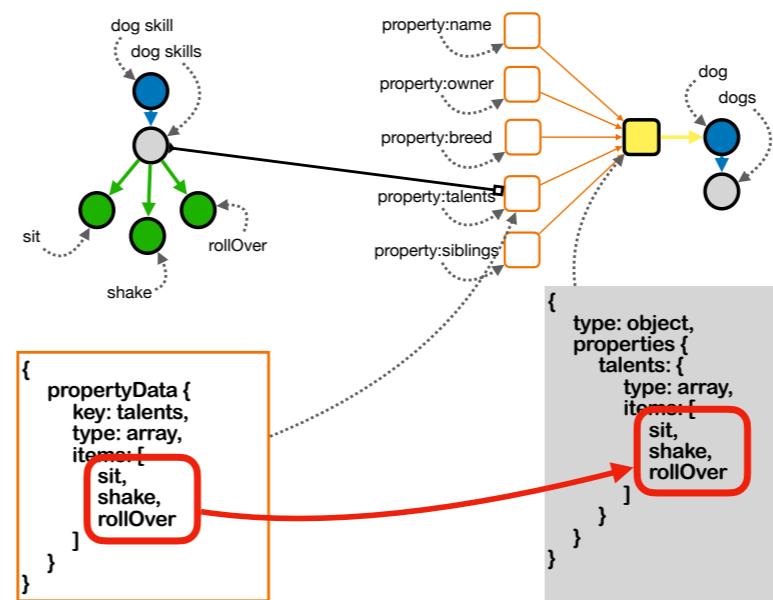
to be entered into items under propertyData (once again, note: a few slides back, we would have added these elements under enum, not under items; that's just how JSON Schema works);

topological encoding of information INTO the graph

front end

properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings



and they are then automatically propagated to the relevant property under JSON Schema.

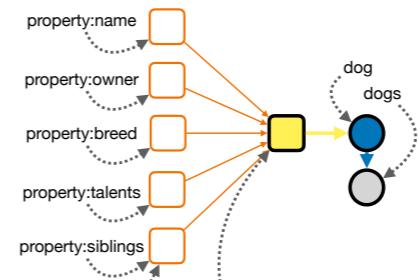
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: siblings,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    siblings: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

We will go through the same process for siblings as we did for talents.

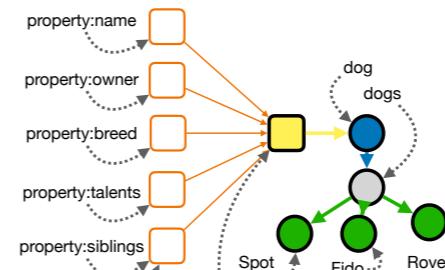
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: siblings,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    siblings: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

The user clicks a few buttons

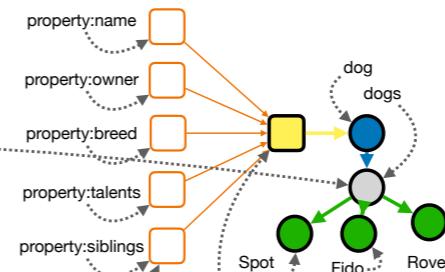
topological encoding of information INTO the graph

front end

+ properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: siblings,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    siblings: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

and tells the CG that siblings should be limited to elements of the concept for dogs.

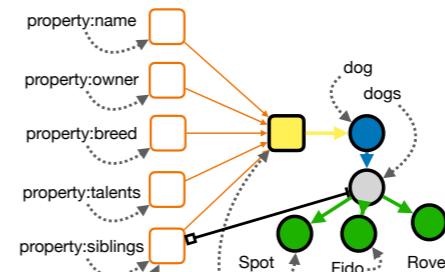
topological encoding of information INTO the graph

front end

+ properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



```
{  
  propertyData {  
    key: siblings,  
    type: array,  
    items: {  
      type: string  
    }  
  }  
}
```

```
{  
  type: object,  
  properties {  
    siblings: {  
      type: array,  
      items: {  
        type: string  
      }  
    }  
  }  
}
```

This triggers a relationship to be created from the node for dogs to the node for property: siblings.

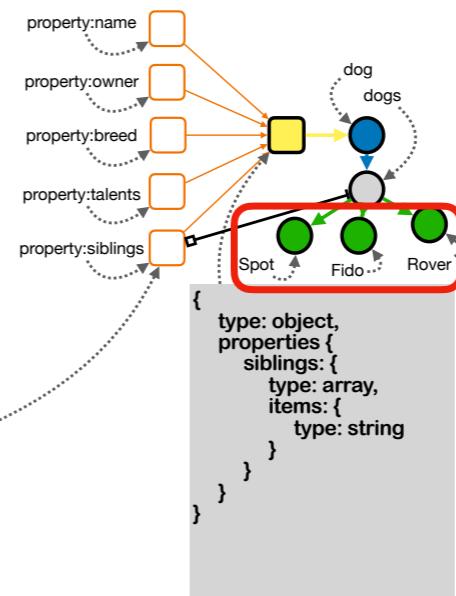
topological encoding of information INTO the graph

front end

+ properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



This relationship induces the elements of dogs (spot, fido, and rover)

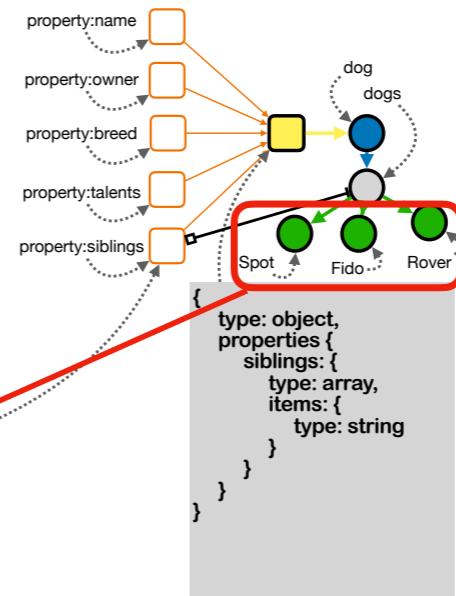
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



to be inserted under `propertyData`, and for the type to be set to array if not already done so.

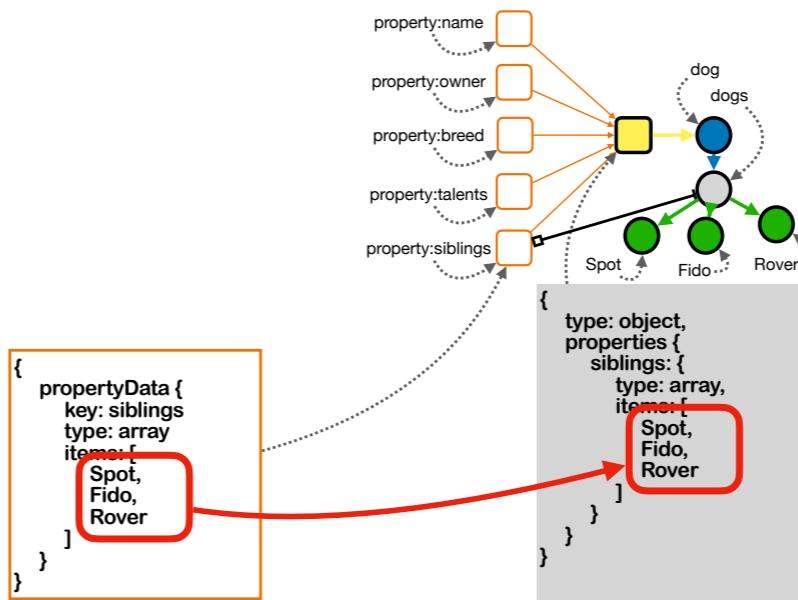
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



These elements then propagate automatically into JSON Schema.

[end 19]

20

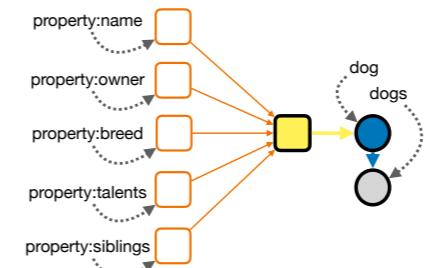
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



[start 20]

Let's say that now, the user looks at the bottom three of these properties

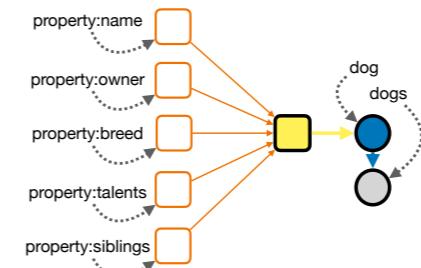
topological encoding of information INTO the graph

front end

+ properties for dog

- ❖ name
- ❖ owner
- ❖ breed
- ❖ talents
- ❖ siblings

back end



and thinks: I want to take these 3 and group them together. Why? bc it pleases me to do so; it makes sense intuitively for me to group them together, and that's all you gotta say.

topological encoding of information INTO the graph

front end

 properties for dog

❖ name

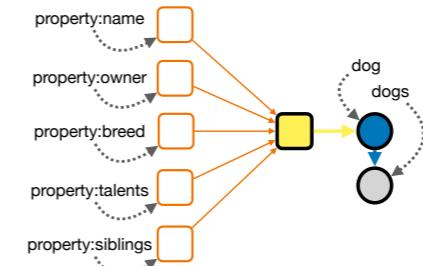
❖ owner

❖ breed

❖ talents

❖ siblings

back end



topological encoding of information INTO the graph

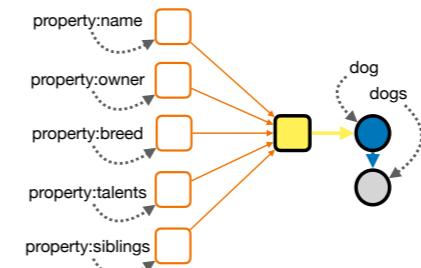
front end

 properties for dog

 name
 owner

 breed
 talents
 siblings

back end



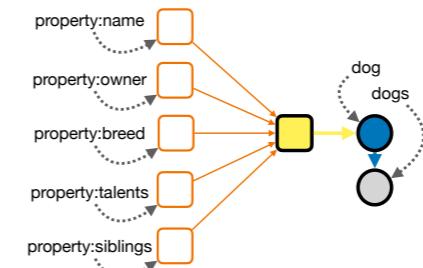
topological encoding of information INTO the graph

front end

 properties for dog

-  name
-  owner
-  pedigreeInfo
 -  breed
 -  talents
 -  siblings

back end



and he thinks: I'll call this grouping pedigree information, bc that's how I think of it. What happens on the back end?

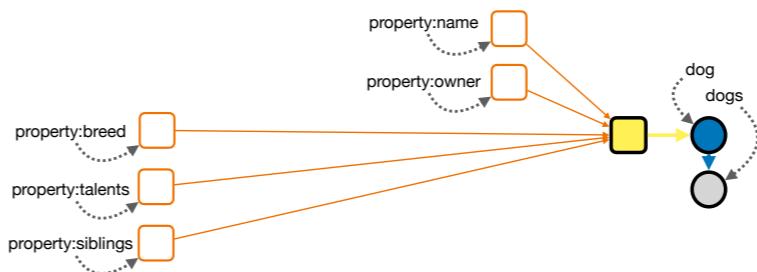
topological encoding of information INTO the graph

front end

 properties for dog

-  name
-  owner
-  pedigreeInfo
 -  breed
 -  talents
 -  siblings

back end



On the back end, the CG takes the three corresponding properties

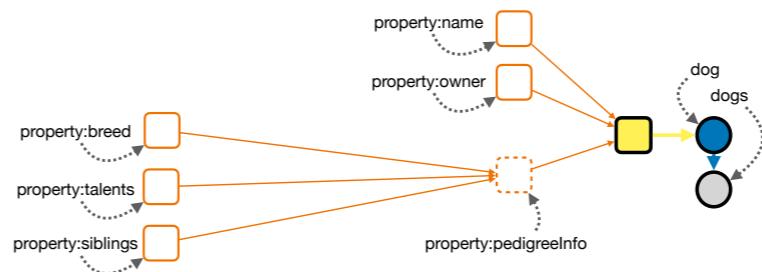
topological encoding of information INTO the graph

front end

 properties for dog

-  name
-  owner
-  pedigreeInfo
 -  breed
 -  talents
 -  siblings

back end



and groups them together into a new property, with key: pedigreeInfo.

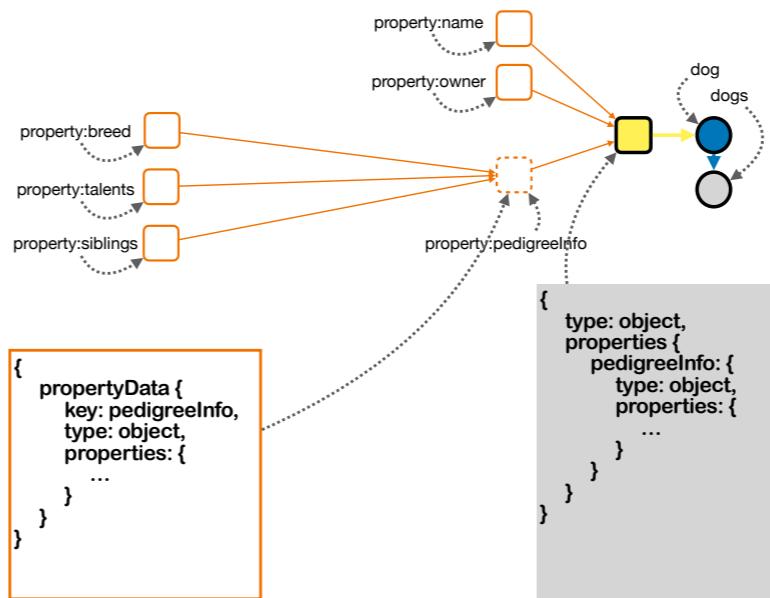
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ pedigreeInfo
 - ❖ breed
 - ❖ talents
 - ❖ siblings

back end



This new property is an object. And even though I'm not showing it here, the three properties would propagate

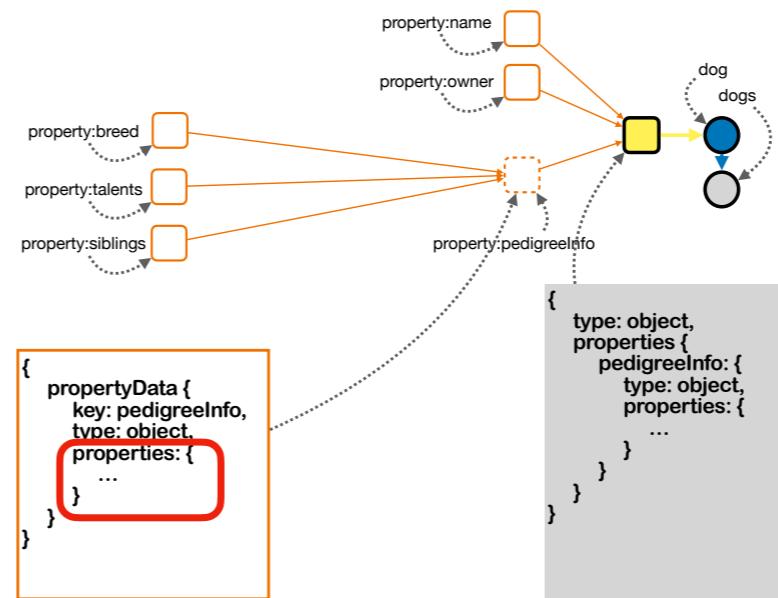
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ pedigreeInfo
 - ❖ breed
 - ❖ talents
 - ❖ siblings

back end



into the file for pedigreeInfo and then propagate from there

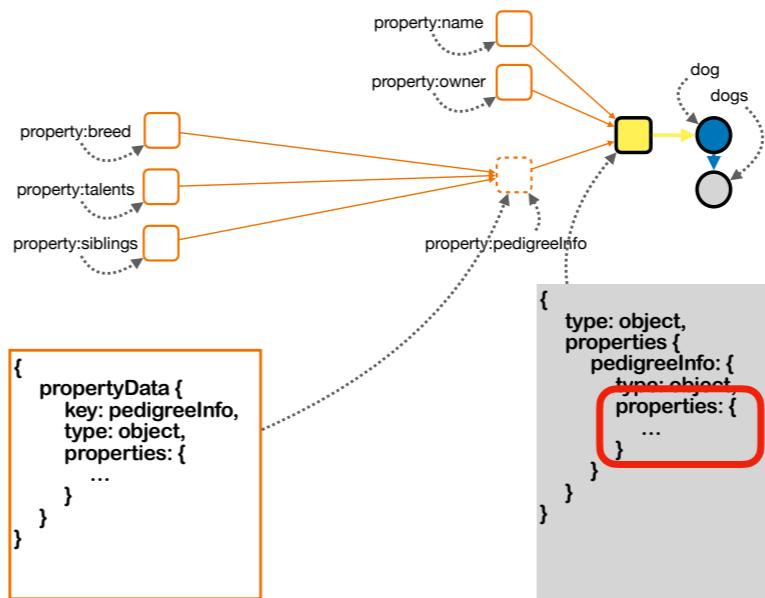
topological encoding of information INTO the graph

front end

 properties for dog

- ❖ name
- ❖ owner
- ❖ pedigreeInfo
 - ❖ breed
 - ❖ talents
 - ❖ siblings

back end



into the JSON Schema on the right.

topological encoding of information INTO the graph

front end

back end

OK, to sum up the points about the CG:

topological encoding of information INTO the graph

front end

- create concepts
- organize concepts into concept graphs

back end

The CG is a tool for users to create concepts and organize them into concept graphs.

topological encoding of information INTO the graph

front end

- create concepts
- organize concepts into concept graphs
- intuitive

back end

The operations on the front end are intuitive, even for a user who is not a developer, does not necessarily even know what a JSON file is.

topological encoding of information INTO the graph

front end

back end

But there is a lot going on under the hood on the back end.

topological encoding of information INTO the graph

front end

back end

- *create / update JSON files*

The CG manages JSON files;

topological encoding of information INTO the graph

front end

back end

- create / update JSON files
- create / update graph database

manages the graph database, which includes relationships between files.

topological encoding of information INTO the graph

front end

back end

- create / update JSON files
- create / update graph database
- update JSON files and graph database based on patterns present in the graph

And it periodically combs through the CG looking for specified patterns, which may trigger an alteration of individual JSON files and / or changes to the CG itself. The number of patterns and the number of updates turns out to be pretty extensive. One of the design principles is to design the operation of the CG so that it doesn't matter what order these updates are implemented. They can happen one at a time; in principle they could be happening in parallel (which is in fact probably how it actually happens in the brain). In its most unsophisticated implementation, the CG code can simply cycle through all patterns and execute any needed updates until there are no more updates to be made.

[end 20]

21

The purpose of Conceptualization

[start 21]

The process of creating a bunch of concepts and organizing them together into a graph is called “conceptualization.”

The purpose of Conceptualization

- **Loose Consensus**

One purpose is bc it is a necessary step in the process of generating Loose Consensus. I am referring to the fact that if you want to ask your grapevine a question, and you want the Grapevine to give you some kind of numerical answer, then depending on what exactly the question is you're probably going to have to make use of at least one concept as you go through the process of designing the question.

The purpose of Conceptualization

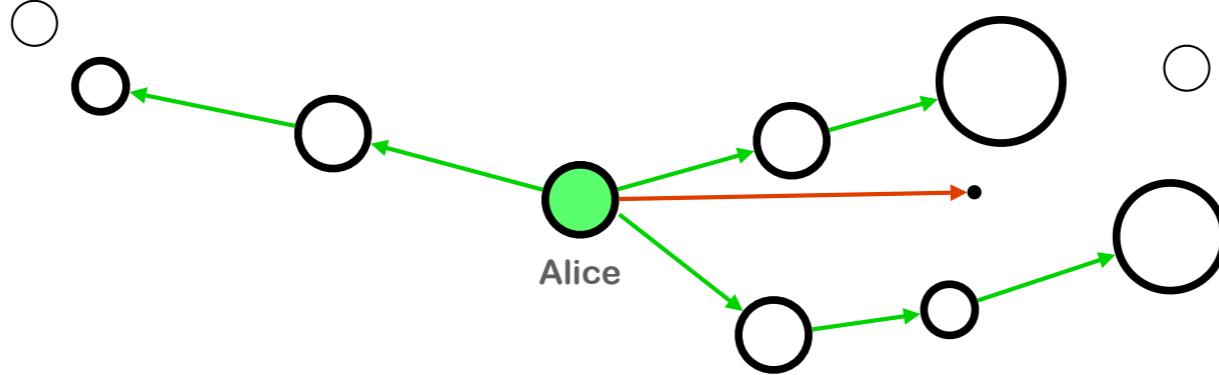
- Loose Consensus
- Should I build it into my app or project?

But there might be other reasons for conceptualizing your app's database, either all of it or just a subset of it. The question now arises: what is the point? Why go through all this trouble to build a concept graph in the first place? Before I explain that, let me review how the Grapevine works. Because the purpose becomes more clear once you think of the two of them as a set of tools that work together.

The Grapevine

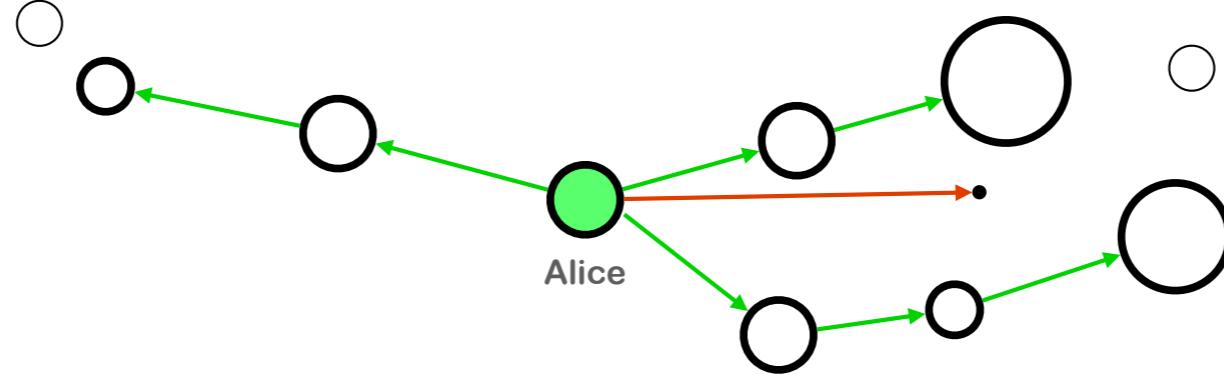
I'd like to review what the Grapevine is and how it works.

The Grapevine



Here we have Alice's Grapevine. Each circle represents a user. (Technically speaking, it is the JSON file for a user.) Each arrow, red or green, represents a rating of one user by another user and is a recommendation regarding how much influence the target user should have in the specified context. Green arrow is a favorable recommendation; red is an unfavorable one. There is a number indicating how much influence is recommended; for the sake of simplicity, I'm not showing that much information on this slide. Based on all available recommendations, Alice calculates an Influence Score for each user; the radius of each circle indicates that influence score.

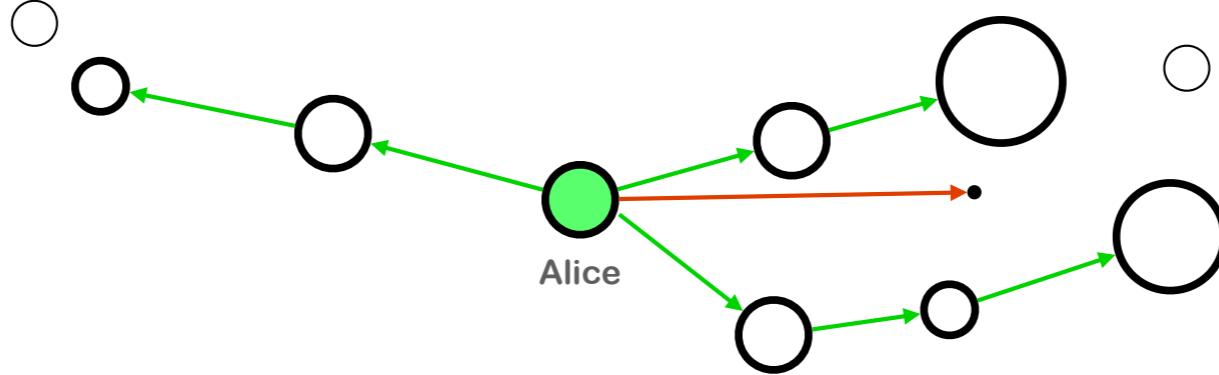
The Grapevine



The first thing to know about the Grapevine is that Influence Scores are contextual, with context being hierarchical. I showed you a Context Tree very briefly earlier in this talk and I'll get back to context trees, which I may also refer to as context hierarchies later. For this slide, imagine we are looking at the Influence Score corresponding to the top of the context hierarchy: context is generic, so we refer to this as “generic” influence.

The Grapevine

**(I'm only showing a small portion of Alice's grapevine.
Her entire grapevine is likely much bigger than this.)**



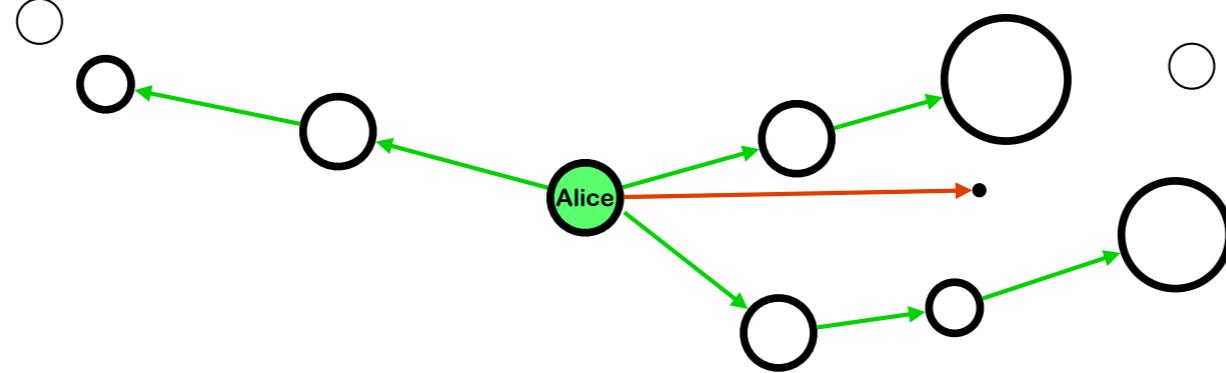
The next thing to know about Alice's Grapevine is that it has the capacity to grow very large. What you are seeing depicted in this slide is only a small portion of her entire Grapevine, for the sake of illustration. How big could it possibly get?

The Grapevine



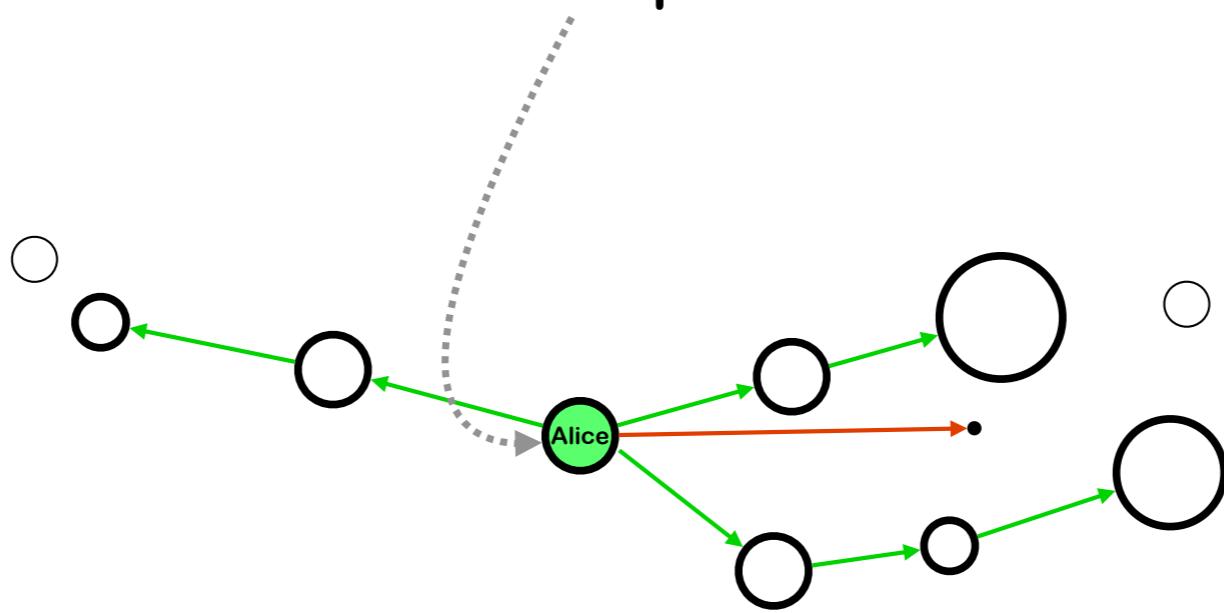
The answer is: This big. It can cover the entire world. Six degrees of separation, and her grapevine can connect her to every other user in the world.

The Grapevine

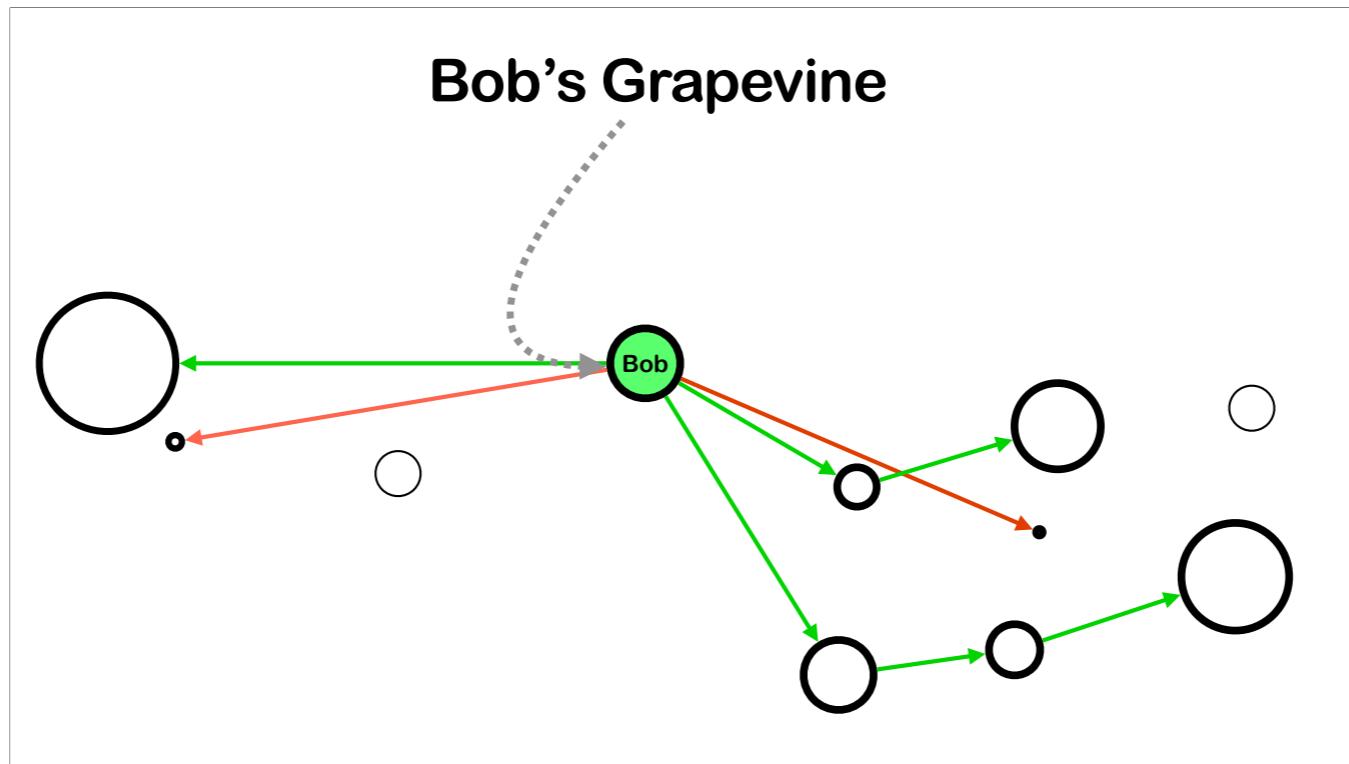


The next thing to know about the Grapevine is that there is no single, universal, grapevine.

Alice's Grapevine

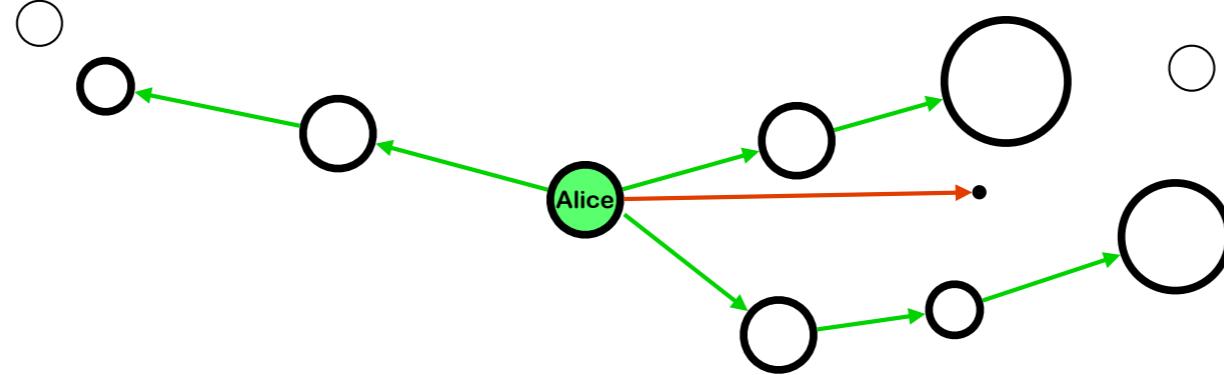


Alice's Grapevine, shown here, is not the same



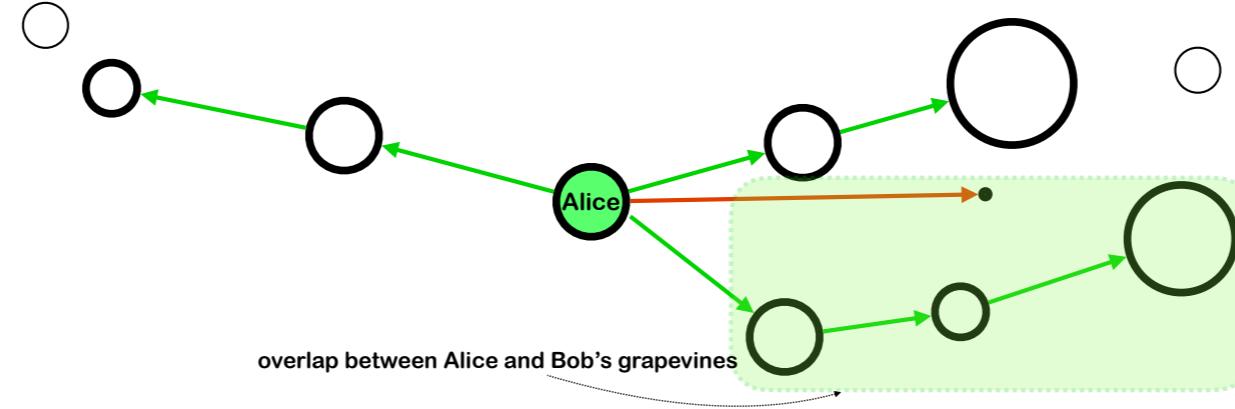
as Bob's Grapevine.

The Grapevine



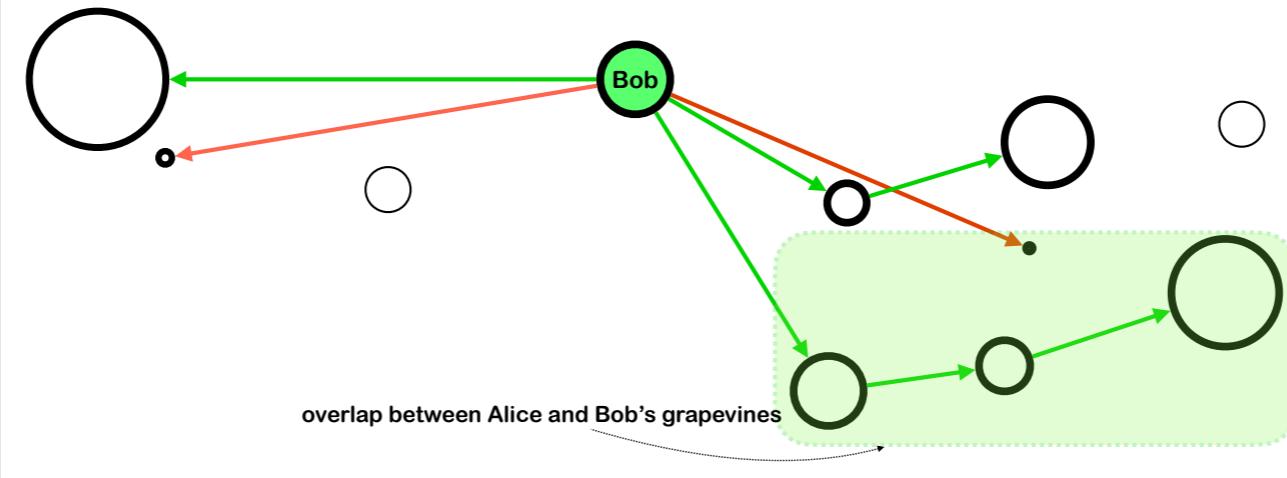
Alice's and Bob's Grapevines would be expected, in general, to be the same in some areas but different in others.

The Grapevine



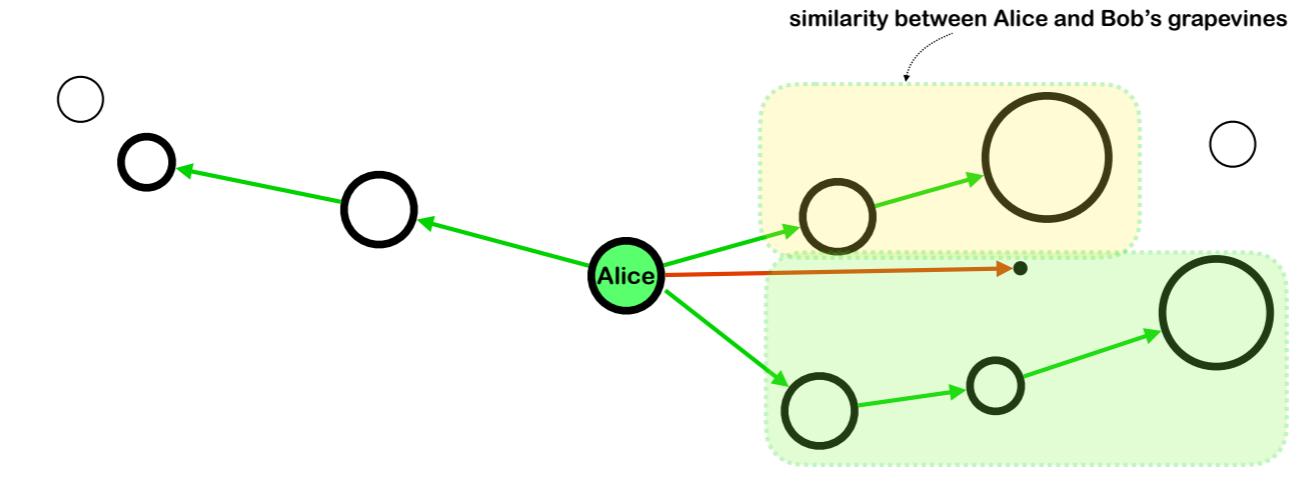
In these slides, the green box represents a subset of Alice's Grapevine

The Grapevine



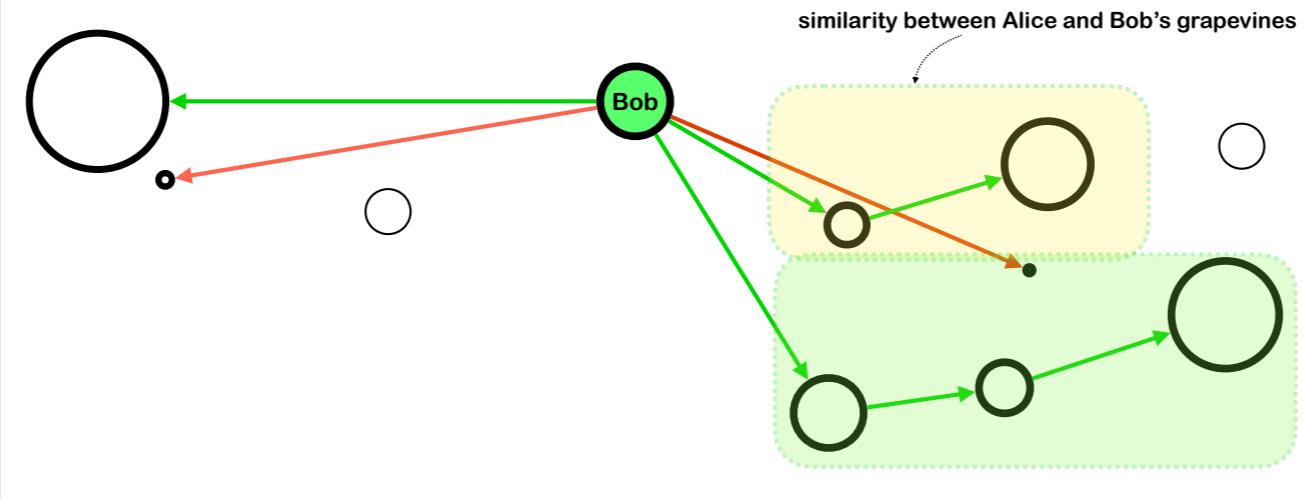
that overlaps exactly with Bob's Grapevine.

The Grapevine



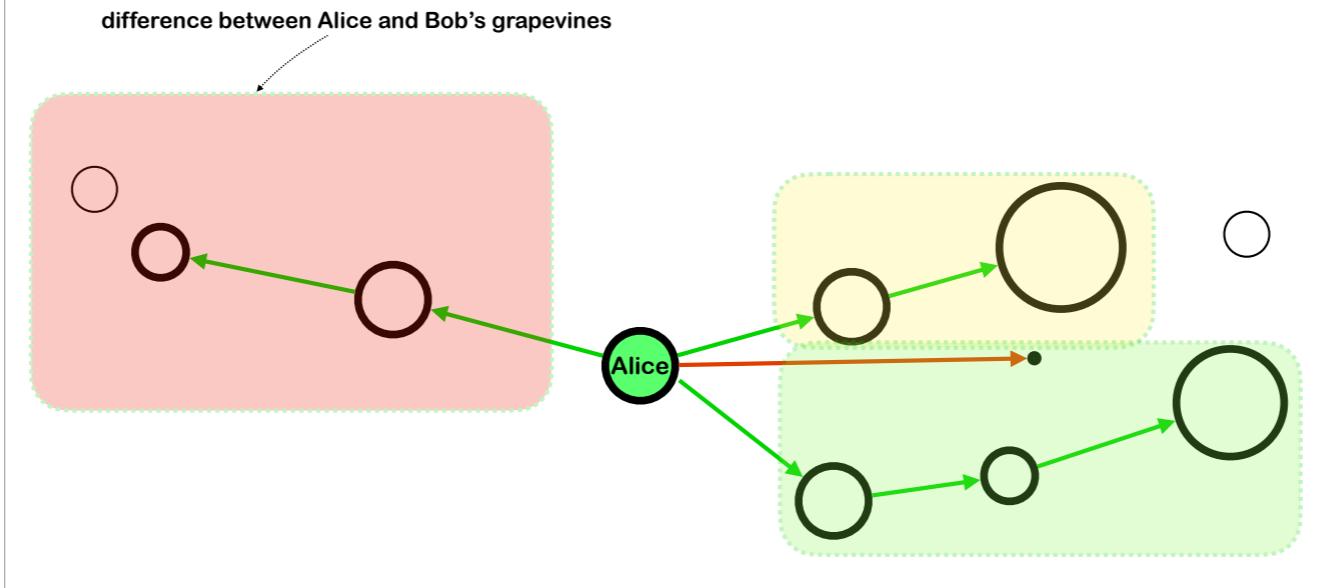
The yellow box indicates an area of Alice's Grapevine

The Grapevine



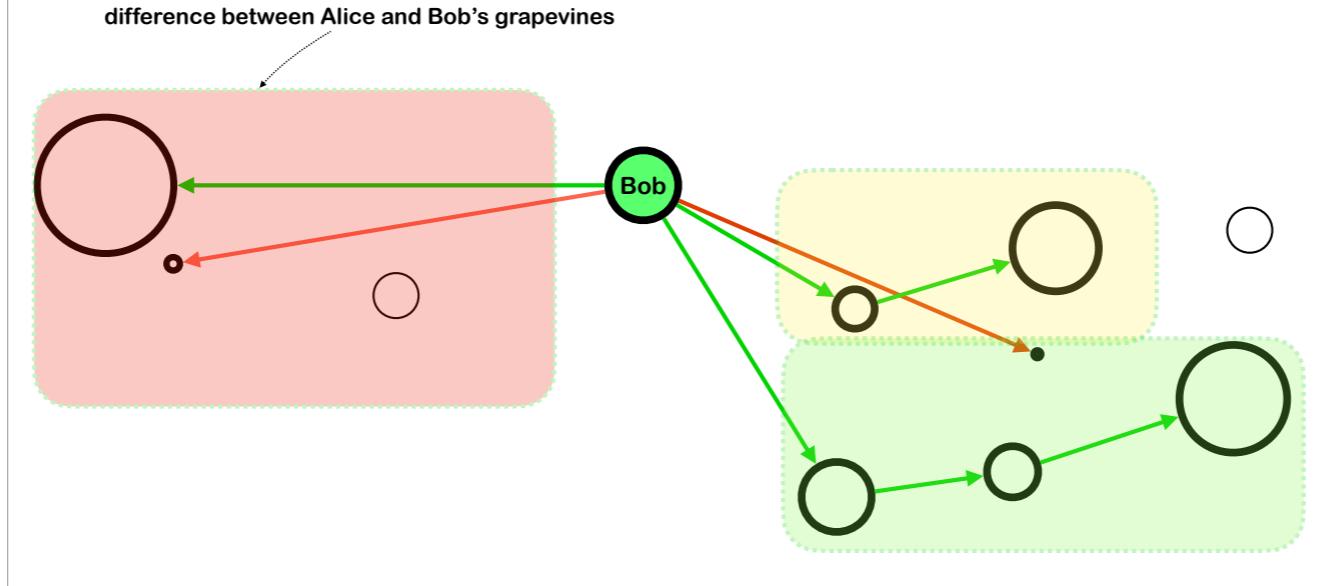
which is similar, although not exactly identical, to Bob's Grapevine.

The Grapevine



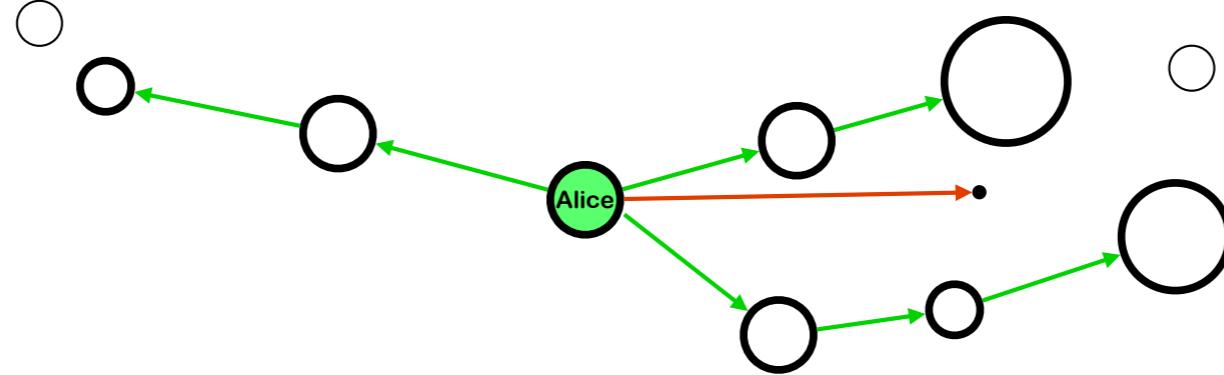
And finally the red box indicates an area of Alice's Grapevine

The Grapevine



that is very much unlike Bob's Grapevine. I haven't shown it here, but there may be some users who are known to Alice but not to Bob; and some users that are known to Bob but not to Alice. And I'm also not indicating it here, but it is quite possible that Bob and Alice may not have exactly identical information about any given user. That's how privacy is envisioned to work in the dWeb: Alice can reveal as much or as little about herself as she so chooses, and to whomever she chooses.

The Grapevine

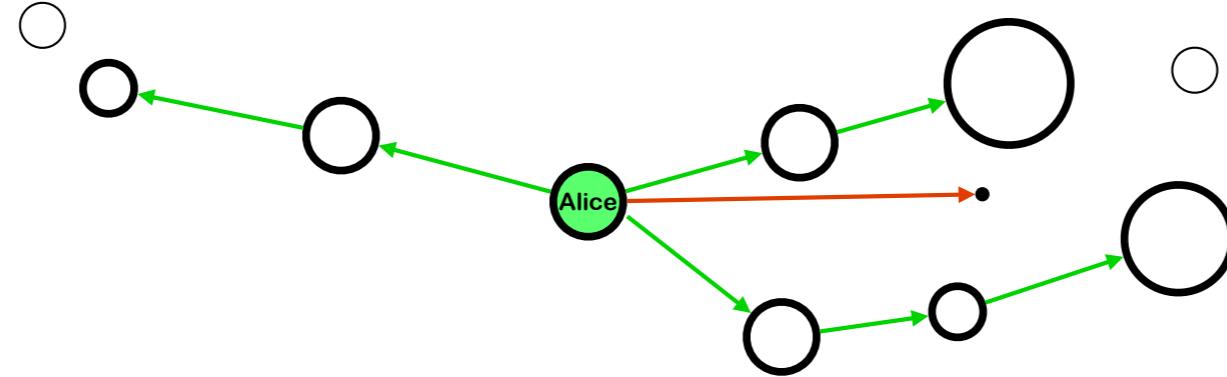


Since their grapevines have considerable overlap, but are probably not exactly identical, we can expect that when Alice and Bob each ask their grapevines to answer the same question, they will sometimes get the same answer, and sometimes get different answers.

[end 21]

22

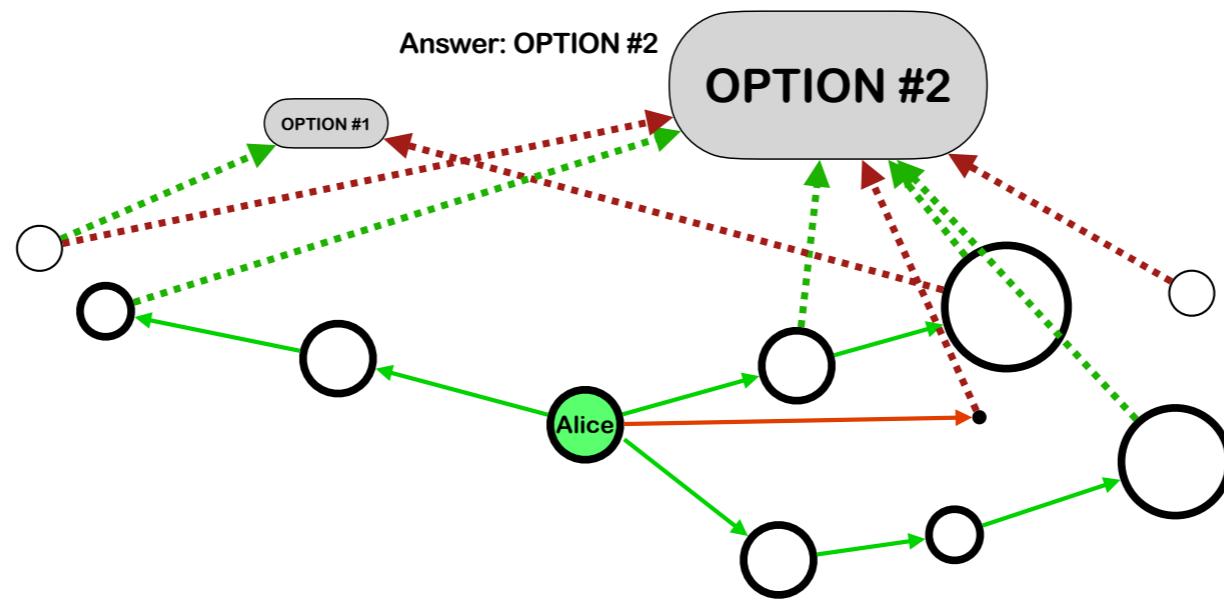
(loose) consensus via the Grapevine



[start 22]

This brings me to the topic of Loose Consensus. Let me walk though how that works.

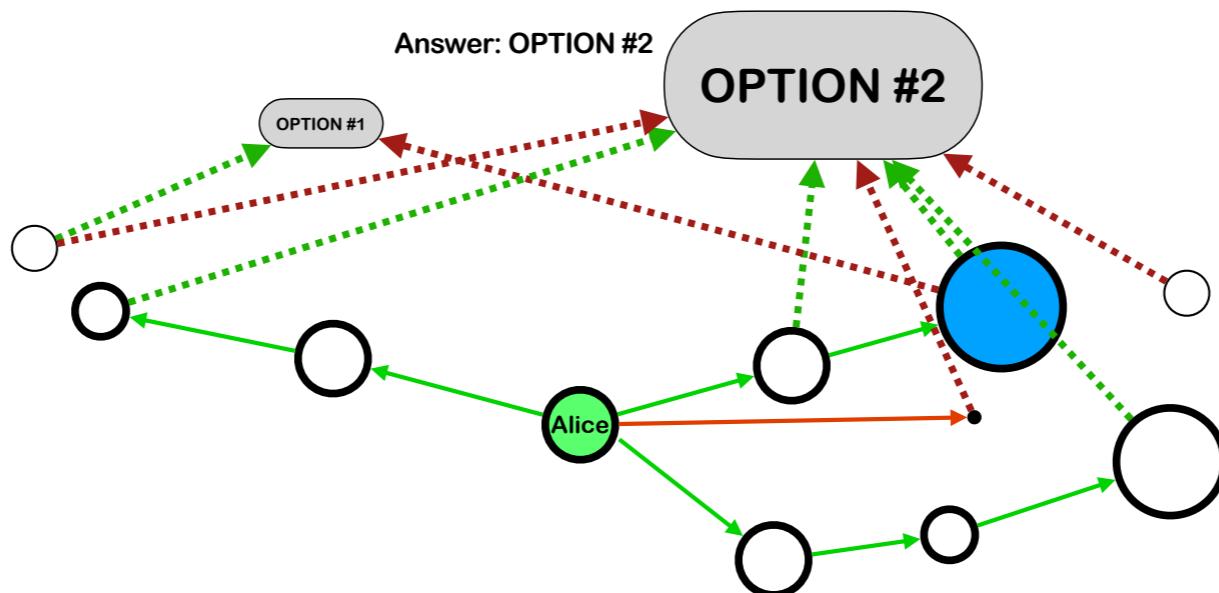
(loose) consensus via the Grapevine



Imagine Alice asks her Grapevine to answer a question, which is to select the best out of several options; for example, option 1 and option 2. These attestations are represented by dotted arrows each of which points from a user to an option. Each of these green dotted arrows from a user to an option indicates the user likes that option; each dotted red arrow indicates that user dislikes that option. Alice's app takes all of these ratings into account, weighting them according to the Influence Score of each user, and produces a score for each option. The sizes of each of the two options indicate their final scores.

(loose) consensus via the Grapevine

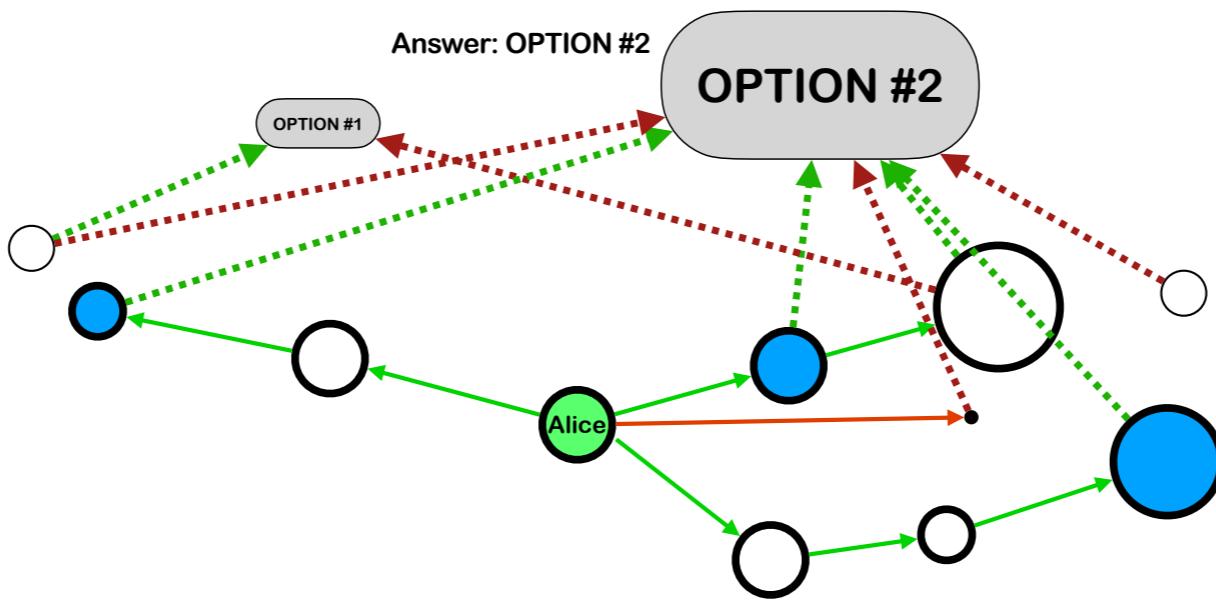
Answer: OPTION #2



For example: the user with the largest diameter, shown in blue, has more influence than any other user on Alice's grapevine. This user likes option two, indicated with the green arrow, and dislikes option 1, indicated with the red arrow.

(loose) consensus via the Grapevine

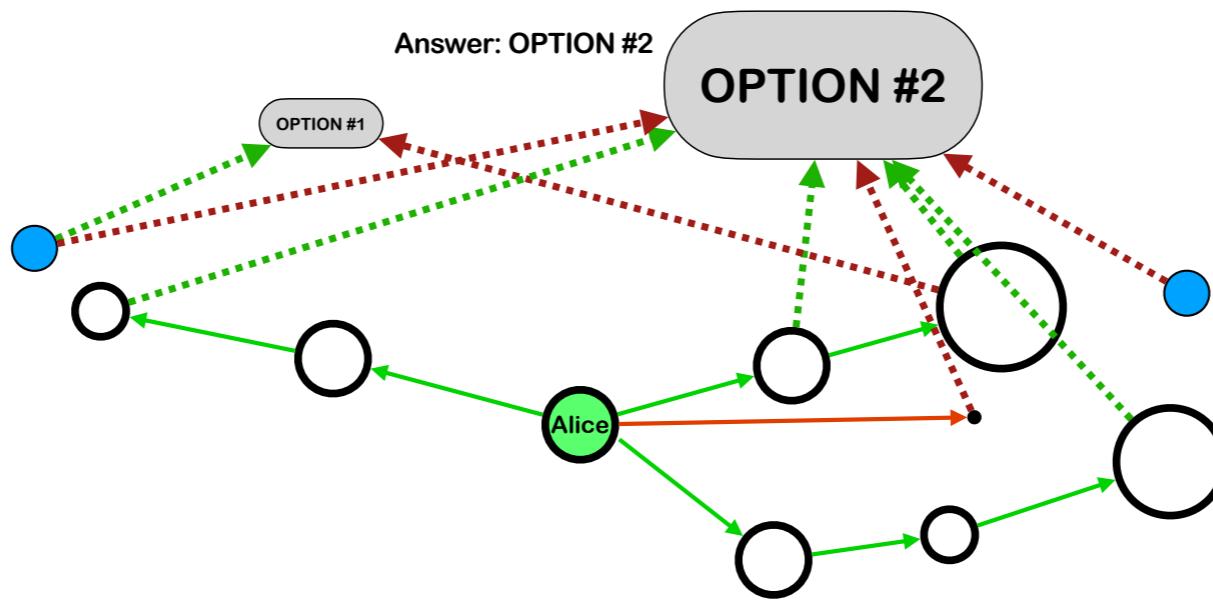
Answer: OPTION #2



These other 3 users, shown here in blue, also like option 2.

(loose) consensus via the Grapevine

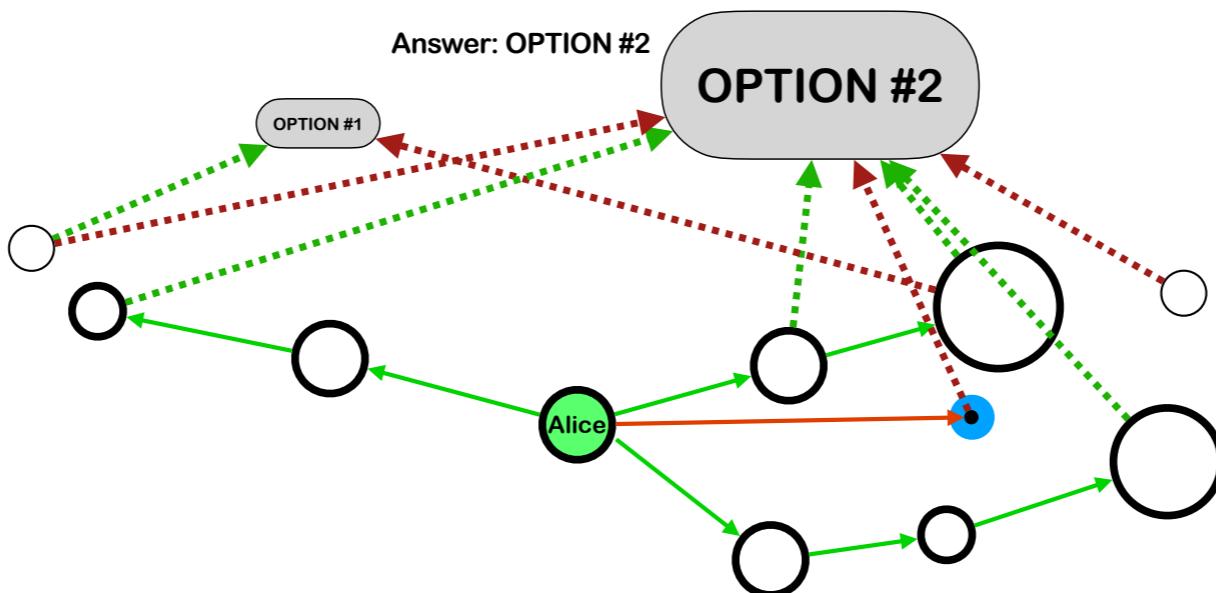
Answer: OPTION #2



The only users who dislike option 2 either are unknown to Alice, and so have not very much influence,

(loose) consensus via the Grapevine

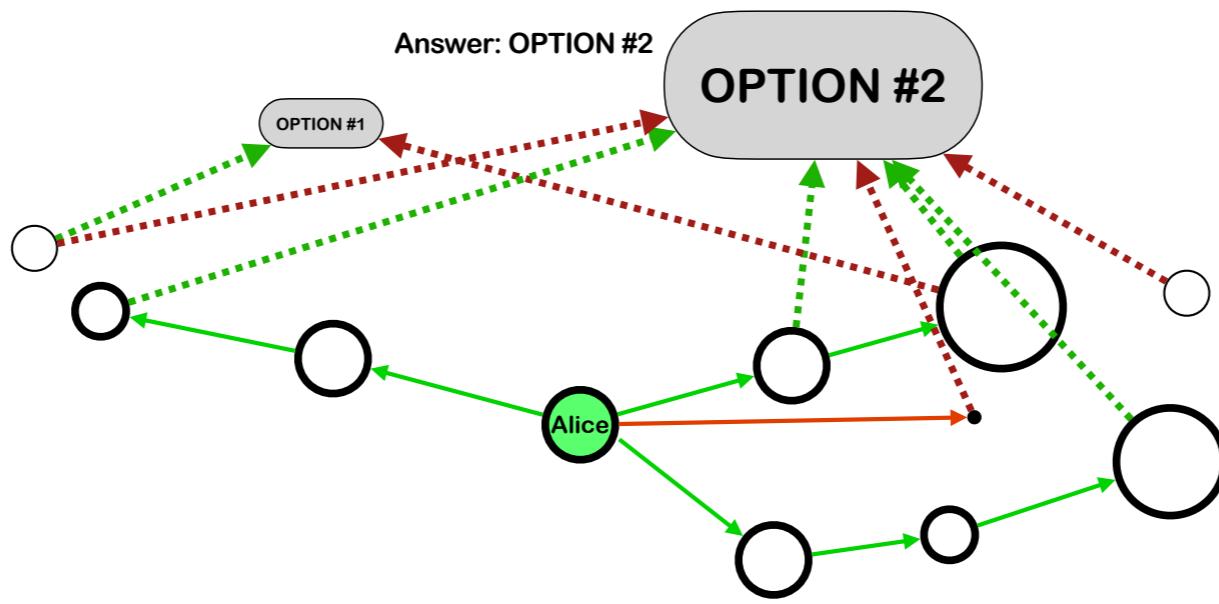
Answer: OPTION #2



or is actively distrusted by Alice, and so has close to zero influence; which is why the diameter of that user is so small, you can barely see it.

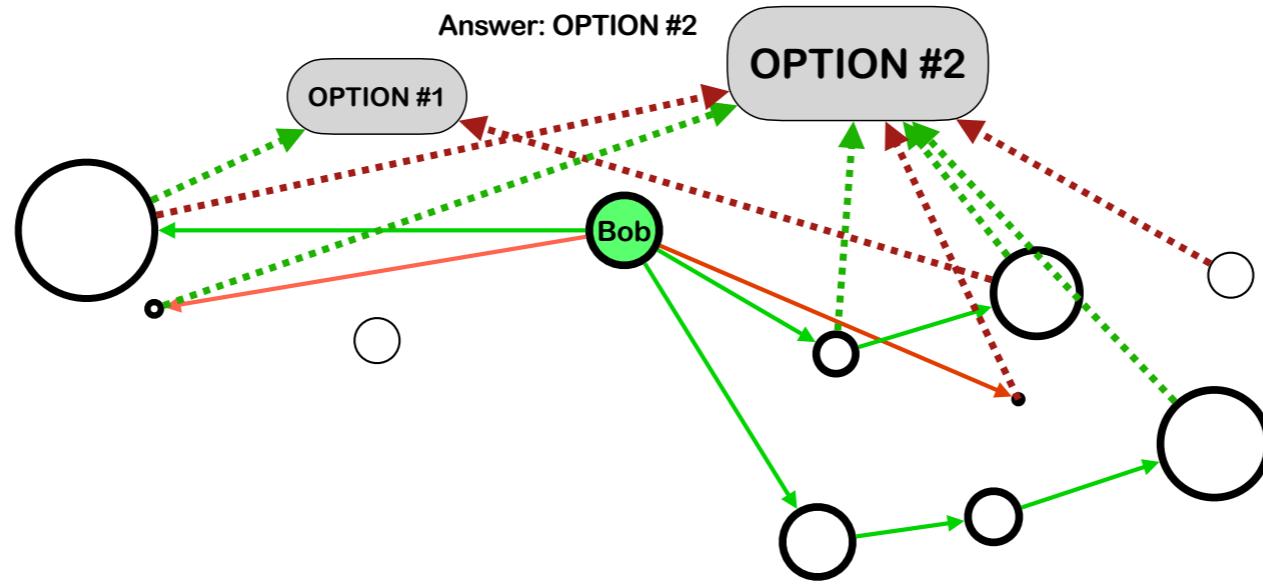
(loose) consensus via the Grapevine

Answer: OPTION #2



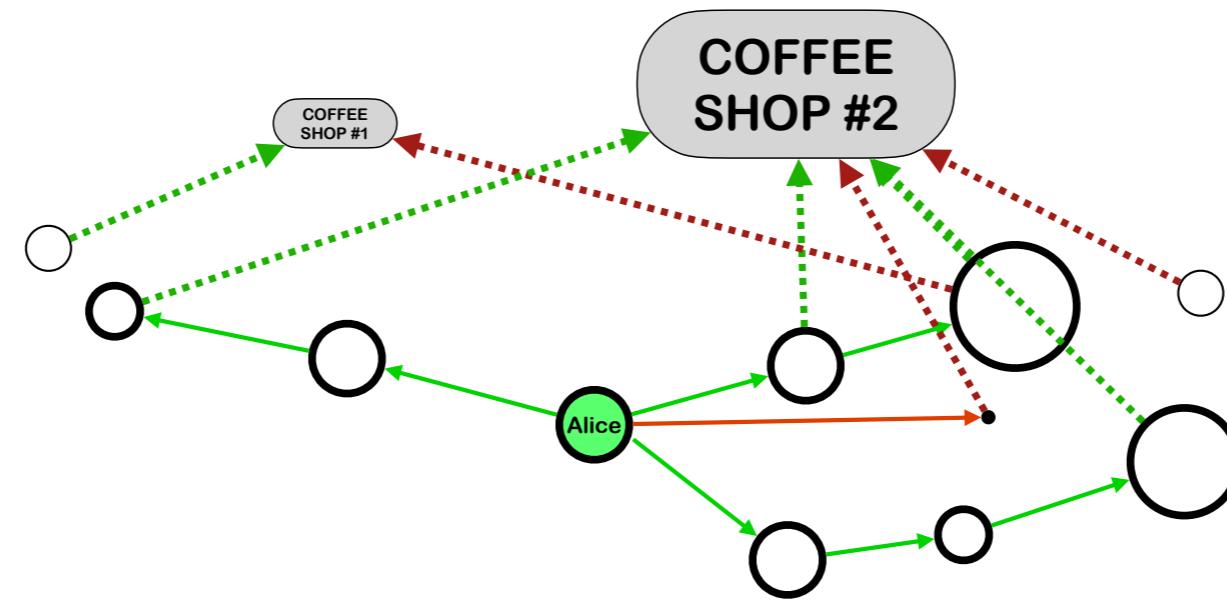
Once all of the numbers are run, Alice sees that her Grapevine favors option 2 over option 1, as indicated on the graph by representing the node for Option 2, top right, to be larger than the node for Option 1, top left.

(loose) consensus via the Grapevine



If we look at Bob's Grapevine, we see that the final scores for options 1 and 2 are a little bit closer; but nevertheless, his Grapevine also picks option 2 over option 1.

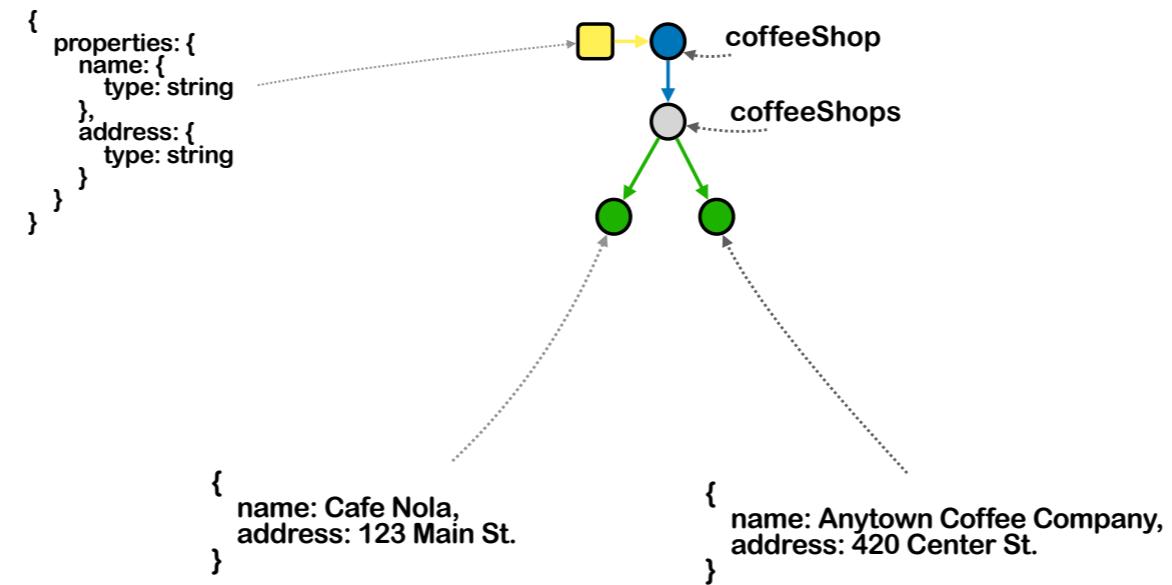
Relationship between the Concept Graph and the Grapevine



The questions that Alice and Bob ask can be about anything. Suppose they want to know the best coffee shop in town. How would they set up this question?

Relationship between the Concept Graph and the Grapevine

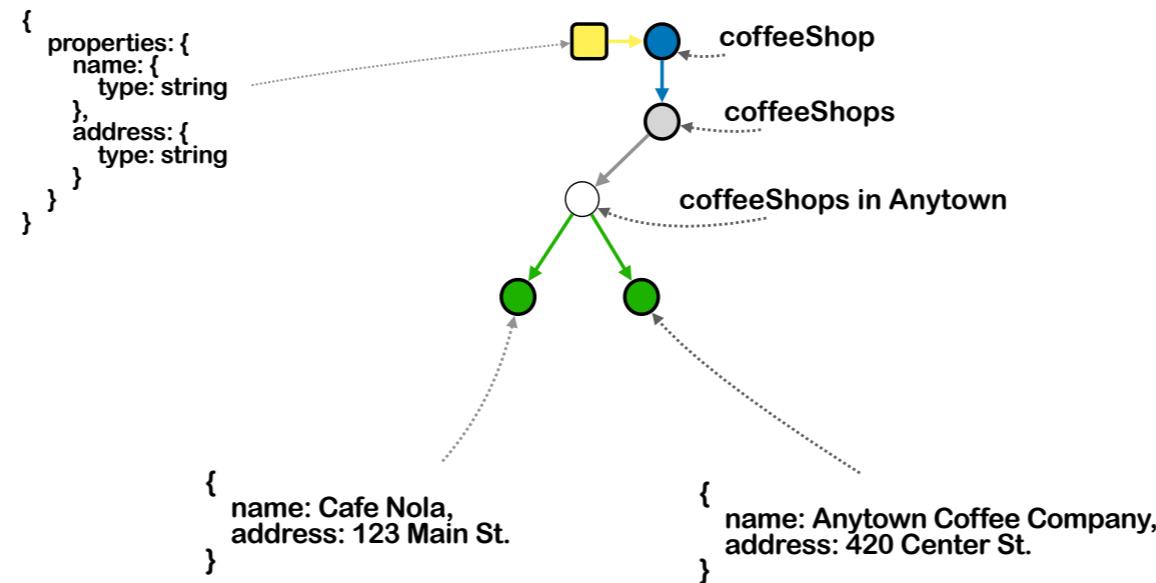
concept: "coffeeShop"



First, they would need to create a concept for Coffee Shops. (Or, more likely to be the case, they would use a concept for coffee shops that has already been created by another user.)

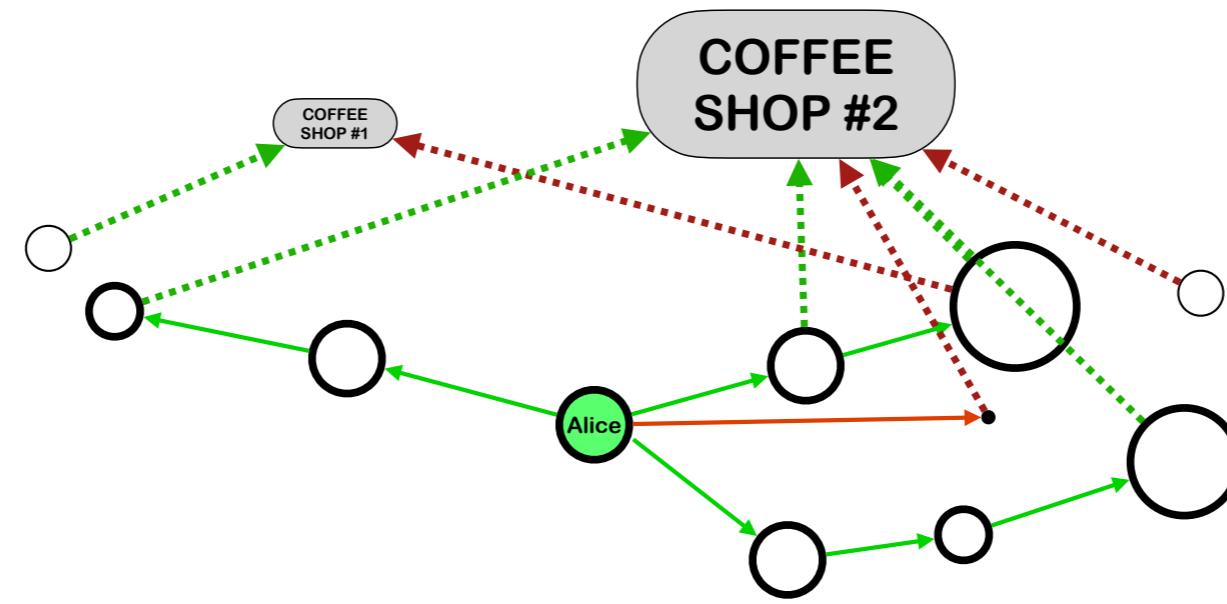
Relationship between the Concept Graph and the Grapevine

concept: "coffeeShop"



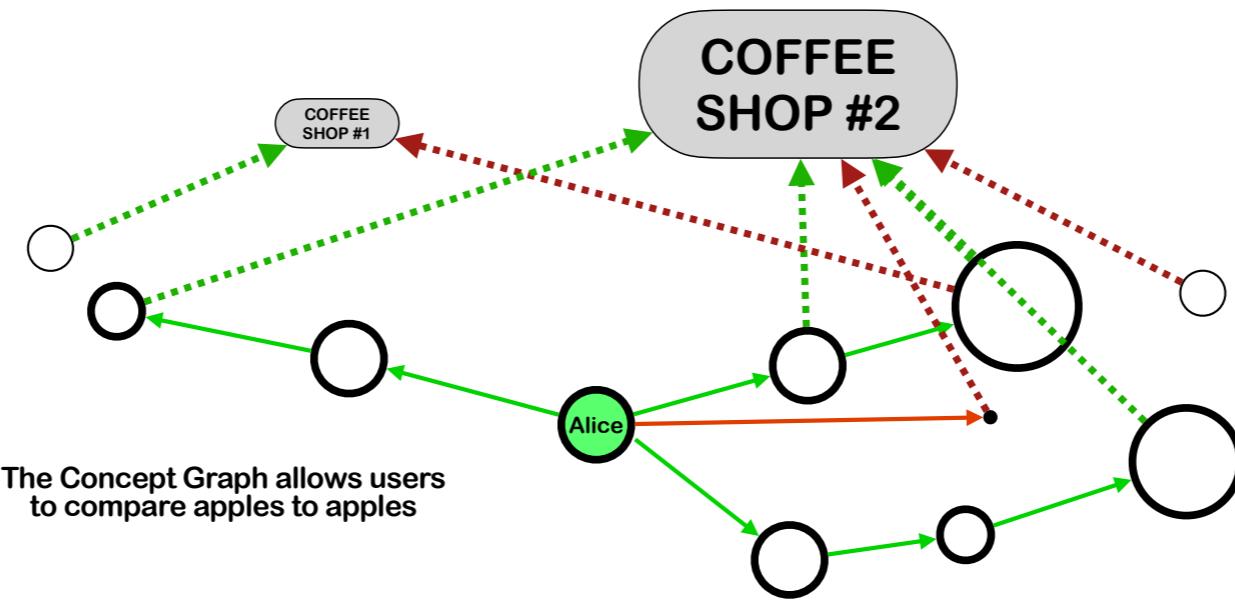
If they are interested in ranking coffee shops in some specific location, like Smallville or Anytown, they would make sure the concept contains a subset corresponding to the location in question. In this example we see the set of all coffee shops in Anytown.

Relationship between the Concept Graph and the Grapevine



Once they make sure the appropriate concept has been created, like coffeeShop, they are able to ask whatever it is they want to ask, like what's the best coffee shop.

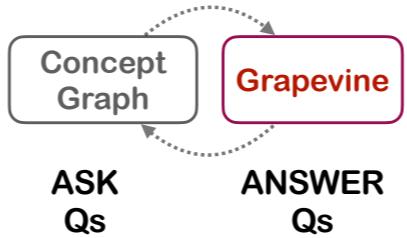
Relationship between the Concept Graph and the Grapevine



In this manner, the Concept Graph and the Grapevine work together to make sure that we are always comparing apples to apples, not apples to oranges.
[end 22]

23

Relationship between the Concept Graph and the Grapevine



[start 23]

So, to sum up: ONE WAY to look at the relationship between the CG and the Grapevine, is that you use the CG to formulate Qs, and you use the Grapevine to answer them.

What kind of questions? Well: they can be on ANYTHING.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

I find it useful conceptually to divide them into 4 categories. Caveat: some questions may fit into more than one category, so these categories may overlap somewhat; but, conceptually speaking, consider these 4 different categories.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

The first category is subjective. There isn't necessarily a right or wrong answer; it is a matter of taste. Questions like: what's the best movie of all time; what's the best tasting burger; and so on.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

The second category is for controversial Qs. Alice and Bob's Grapevines will never agree on everything, most definitely things that are controversial. And that's OK. Freedom to disagree is part of the cypherpunk ethos of decentralization. This category will never go away, nor should it.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

Third category, is an important category: objective fact, shown here in green. Did something happen. Did Country A invade Country B on such and such a date. This is important bc when information is filtered through large powerful entities that lack adequate checks and balances, whether they be governmental entities or corporate or whatever, we can sometimes find ourselves in a situation where we're not sure what's true and what's not. I believe the decentralized web can be a powerful tool to help fix that.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

The fourth category is the most interesting; maybe even more important than the “objective fact category,” it’s the LOOSE CONSENSUS category, shown here at the top, in purple.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

answers are arbitrary
but consensus is required
("social constructs")

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

These are the questions whose answers are what some people might call SOCIAL CONSTRUCTS. Their answers may be arbitrary, meaning there's no single universally right answer; but consensus is of paramount importance. Things like: how are we going to format a JSON file for a user on decentralized Twitter app. Or how are we going to format: whatever. These questions may be tedious, full of details, boring, mundane, technical, NOT controversial (necessarily; sometimes they may be, but typically not really); but questions on which we MUST have CONSENSUS if we are going to be able to communicate, if we are going to tackle the problem of WALLED GARDENS.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

answers are arbitrary
but consensus is required
("social constructs")

OBJECTIVE / FACT:

- did X happen

disinformation 

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

The objective fact category ... That's how I think we're gonna attack disinformation, or what some would call Fake News. That, to put things mildly, is an important problem to tackle.

“Loose” consensus via the Grapevine

TECHNICAL / BORING / MUNDANE:

- What's the best way to FORMAT a user file?
- What's the best way to FORMAT a rating file?
- What's the best way to FORMAT a message file?
- What's the best software package for the Grapevine?

OBJECTIVE / FACT:

- did X happen

SUBJECTIVE / OPINION:

- What's the best movie of all time?
- What's the best burger joint?
- What's the best bitcoin wallet?
- What's the best front end for the decentralized Twitter app?

CONTROVERSIAL:

- Who should be the next President?
- Is Craig Wright the real Satoshi? (lol)

~~Walled
Gardens~~

~~disinformation~~

The top category, in purple; technical, boring, mundane, questions on things like how to format data; Social Constructs ... That's how I think we're gonna attack Walled Gardens. Hard for me to say which of these two problems is more significant. But certainly: both of these problems urgently require solutions.

Many of us have believed for many years that the dWeb should be able somehow to tackle these problems; we haven't yet figured out how. We haven't yet made it happen. We've been missing something. Well, I believe THIS is what we've been missing. Principle of Loki, CG, Grapevine, Loose Consensus. These are what we need to build.

[end 23]

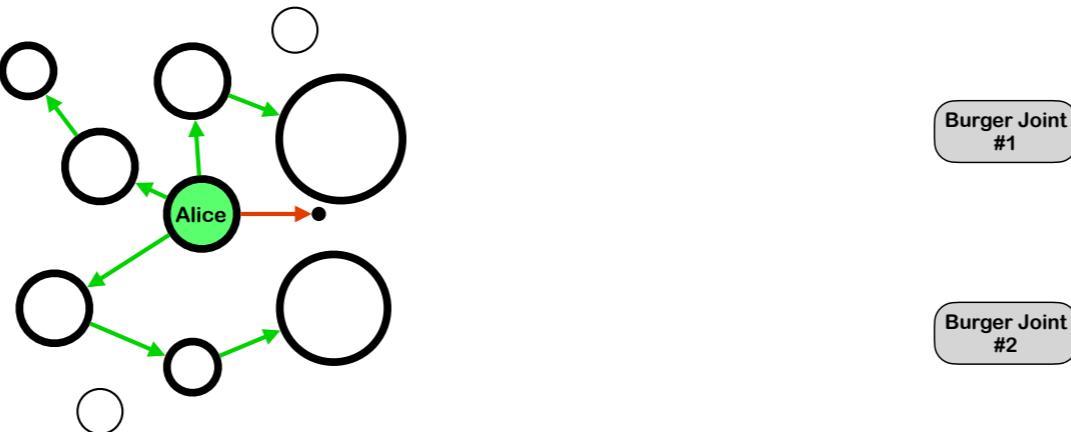
24

Micropayments, monetization, and overturning today's Advertising-based business model

[start 24]

Just to drive home, just how transformative I think these can be: ... Consider that today's web of proprietary platforms, web 2.0, is monetized to a great extent through advertising. I believe the advertising model is soul-crushing and needs to be destroyed. The advertising model is the impetus behind algorithms that are designed to manipulate our emotions, monopolize our attention, and keep our eyes glued to our screens. Good for advertisers, but NOT what we had in mind as users when we started to use social media; it's not fundamentally what any of us really want for ourselves. The thing is: we've simply never known any other way. We are like fish, who have no concept of water, no knowledge of air. Many of you are probably saying to yourself right now: It's just the price we pay for free stuff. And you're OK with it. Well I think that's bc users have no knowledge of any better alternative. YET.

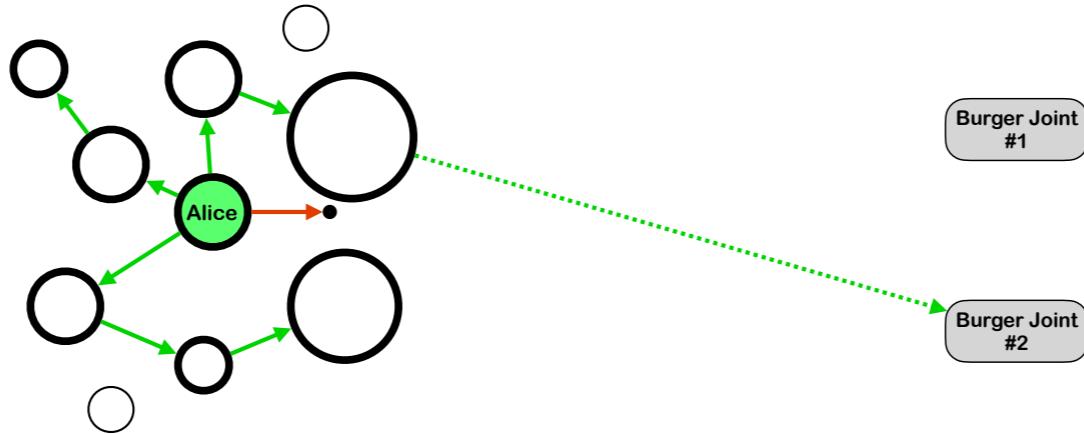
Micropayments, monetization, and overturning today's Advertising-based business model



The system I am describing is ideally suited to provide a better alternative.

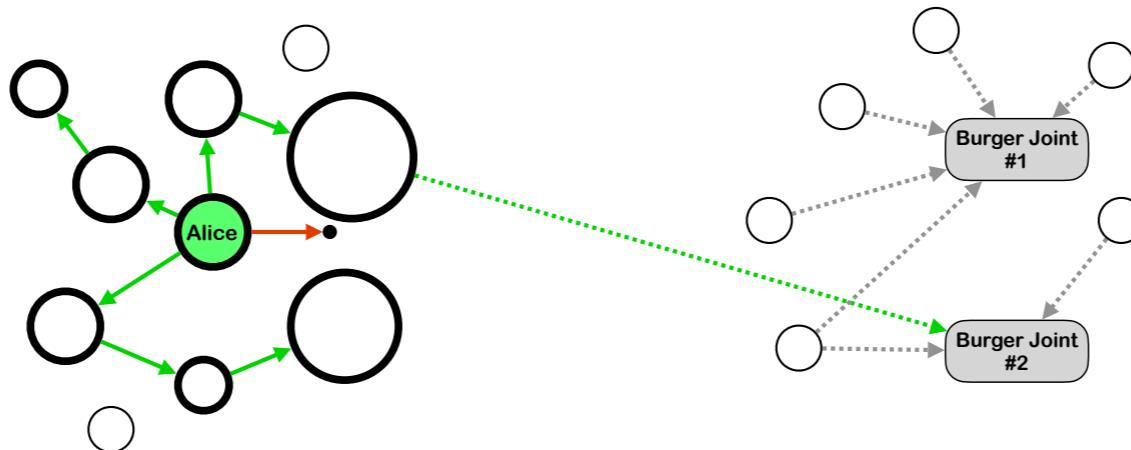
Imagine a system where you can leave a rating; digitally signed, an attestation — of another user, a business, whatever — it can be subjective opinion, burgers here are awesome, or objective fact, like I saw roaches in the food, or they are closed on Tuesdays, either way — and you can set it up so that access to your attestation is free, OR you can charge a microfee, let's say, 100 satoshis, payable over lightning; or peg it to be equal to some fiat amount, but let's say, a nickel's worth of sats, for anyone who wants to get an unencrypted copy of your attestation. Payment can be handled by your app, over the lightning network, in batches so you can OK them all with one click of one button. To make things easy, you set your app so that you will purchase attestations from other users only if the price is below some reasonable amount, and only if they are authored by users who are trusted by your grapevine above some certain threshold, and you'll never spend more than some preset number of sats on ratings, just in case something weird happens, and a zillion attestations become available.

Micropayments, monetization, and overturning today's Advertising-based business model



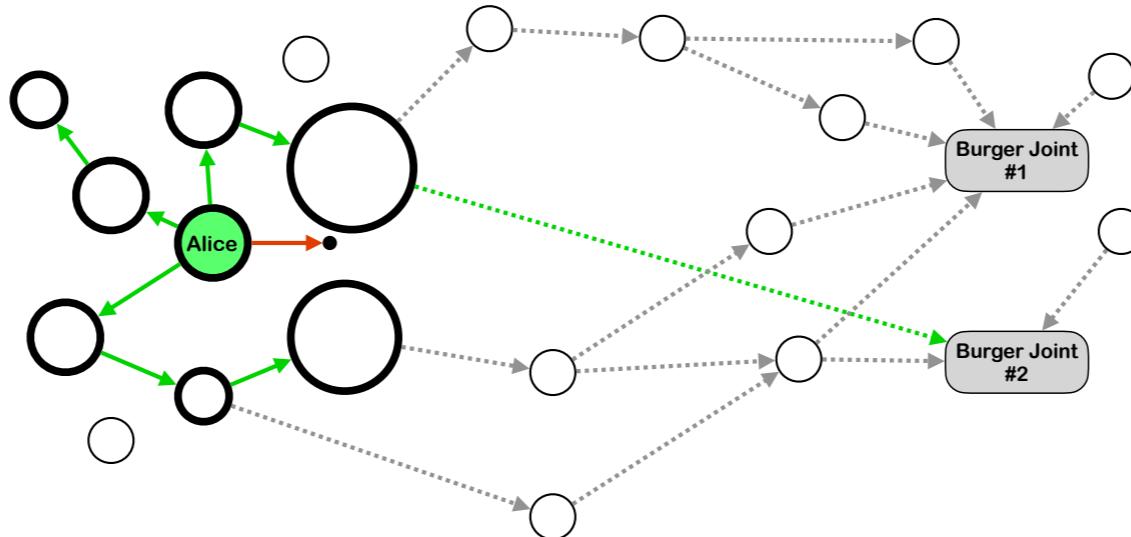
You go to a new town. You want to get information on the best burger joint. You canvas your Grapevine [IOW the users in your user database, which is stored on your device locally] and see one of them left a rating. And it's right there in your database. Awesome. But: you want more than just one rating.

Micropayments, monetization, and overturning today's Advertising-based business model



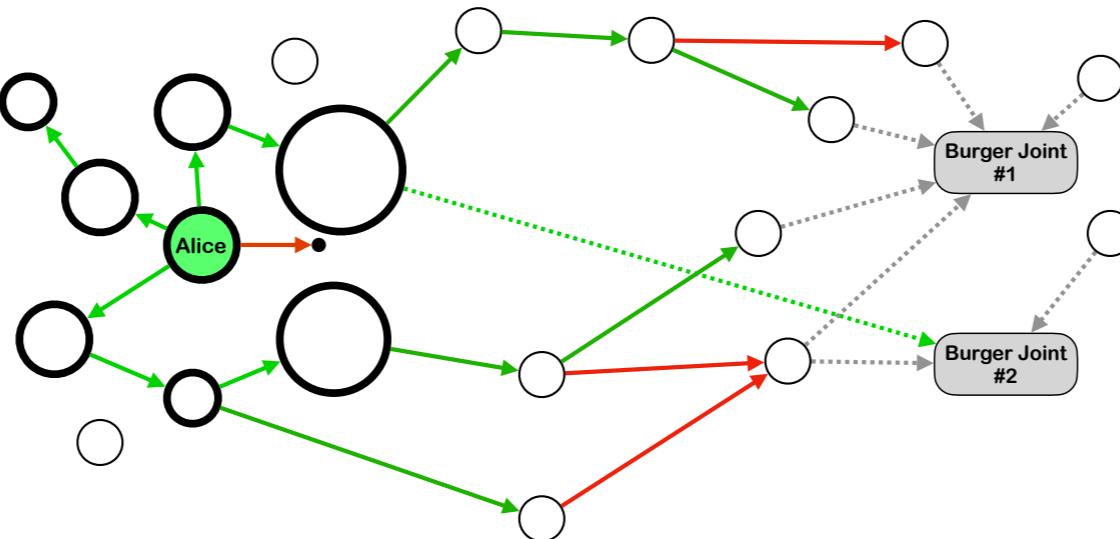
Your device sends out a request through a decentralized network for more information. The burger joints, or perhaps some third party, store and reveal to you the fact that there are encrypted attestations from users who are unknown to you. You don't know yet what the attestations say, bc you haven't OKed the attestations purchase yet. So I show them here in grey, to indicate the number of stars, or whether it's thumbs up or thumbs down, or whatever, is not yet known to you.

Micropayments, monetization, and overturning today's Advertising-based business model



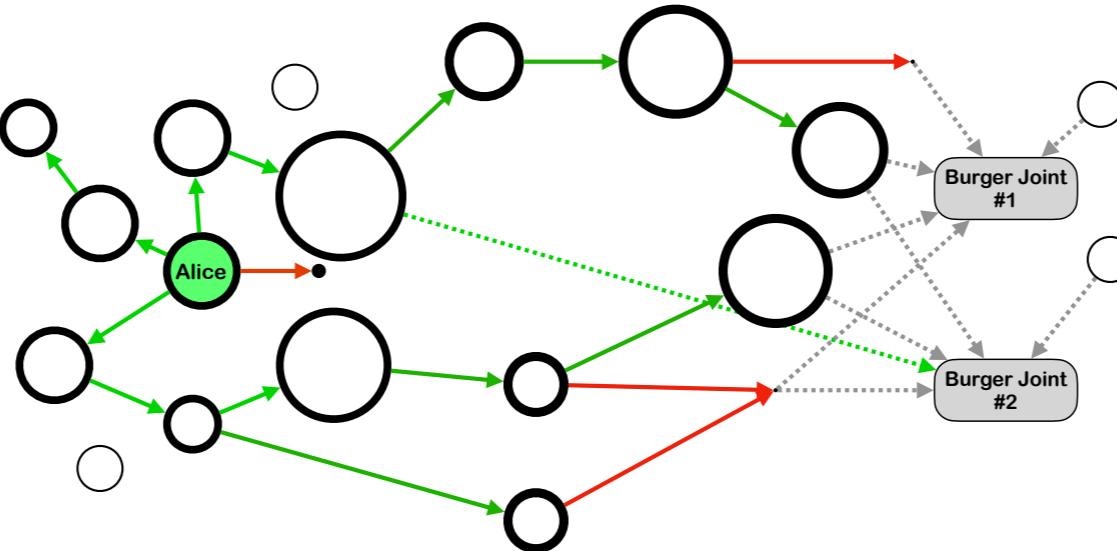
Luckily, after some more searching through the 6-degrees-of-separation, your app discovers connections between you and most of these other users. Depicted here in dashed grey arrows are user-to-user rating attestations that you know exist, but you have to purchase for a few pennies each in order to see what they say. OK. You click a single button and then pay for them all, out of your Lightning-enabled BTC wallet. (Or perhaps it's Filecoin over IPFS. Whatever people build.)

Micropayments, monetization, and overturning today's Advertising-based business model



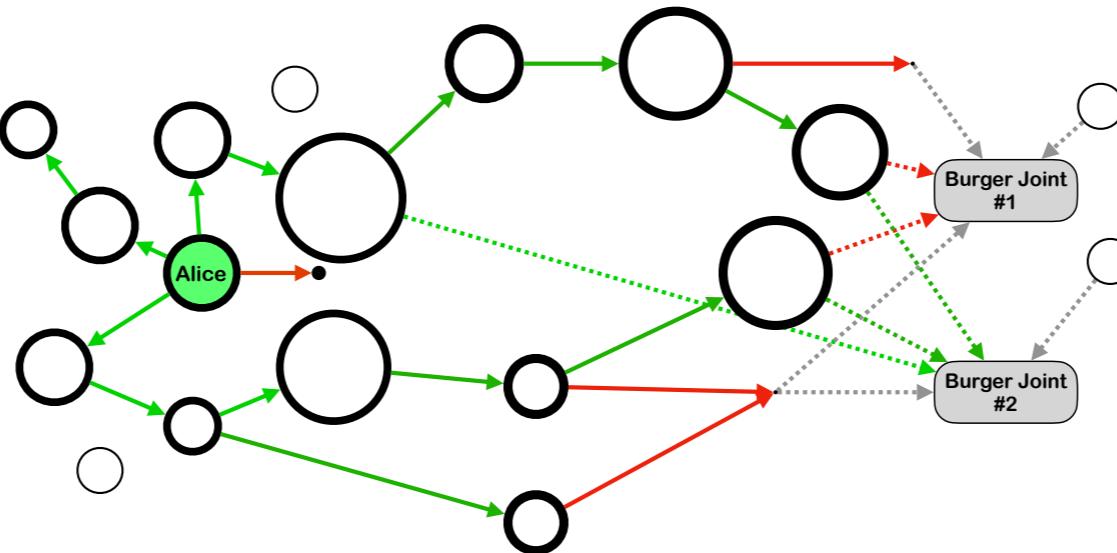
You pay the microfees, your app calculates user influence scores,

Micropayments, monetization, and overturning today's Advertising-based business model



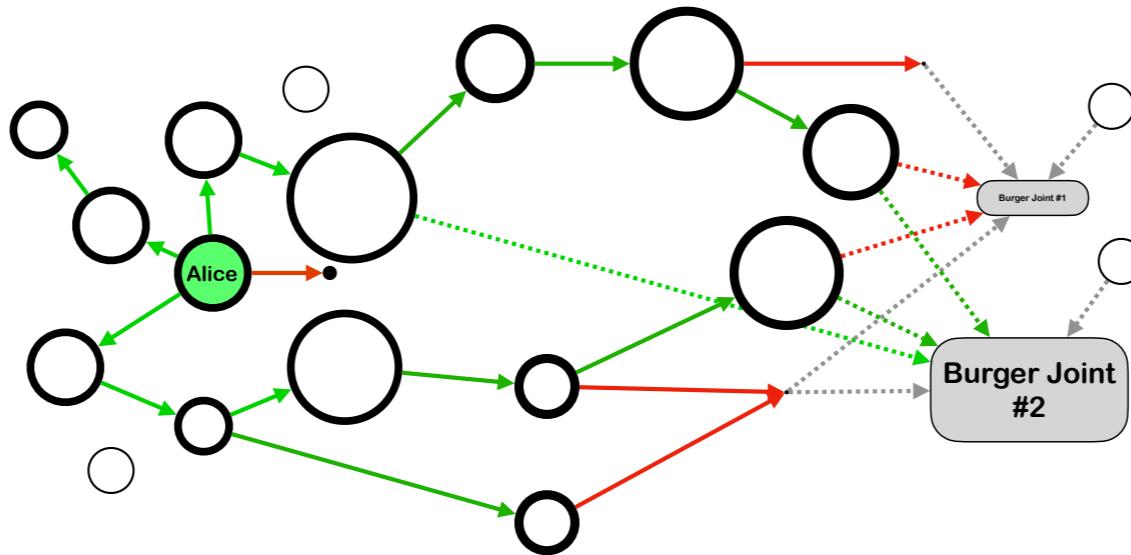
and you see that some of the users are trustworthy, a few others are NOT; two of them, in fact, their calculated influence score is practically zero; and you decide to purchase ratings of the burger joints from the trustworthy users, not the untrustworthy ones.

Micropayments, monetization, and overturning today's Advertising-based business model



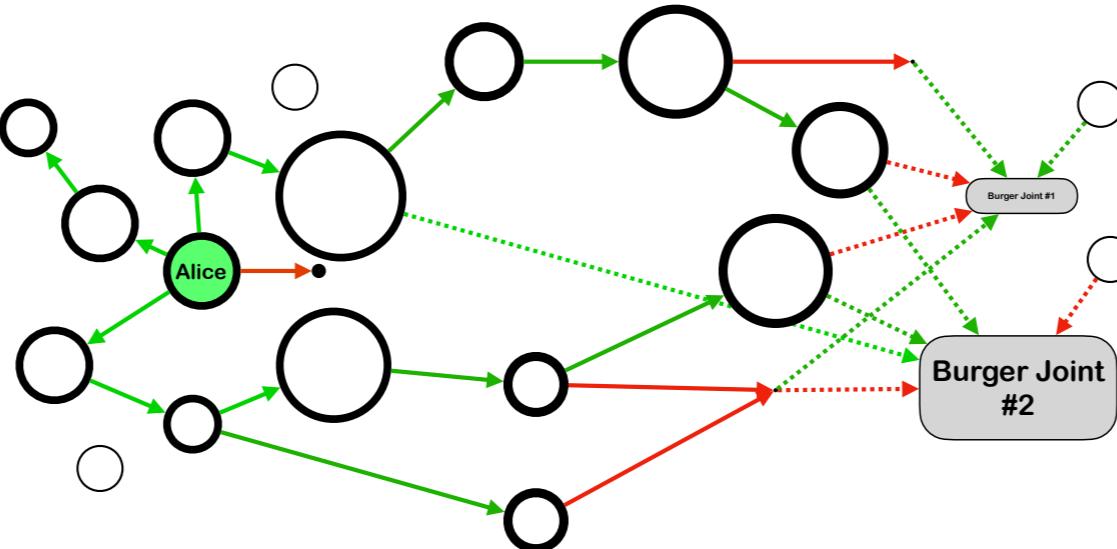
You get the results, your app runs the calculations,

Micropayments, monetization, and overturning today's Advertising-based business model



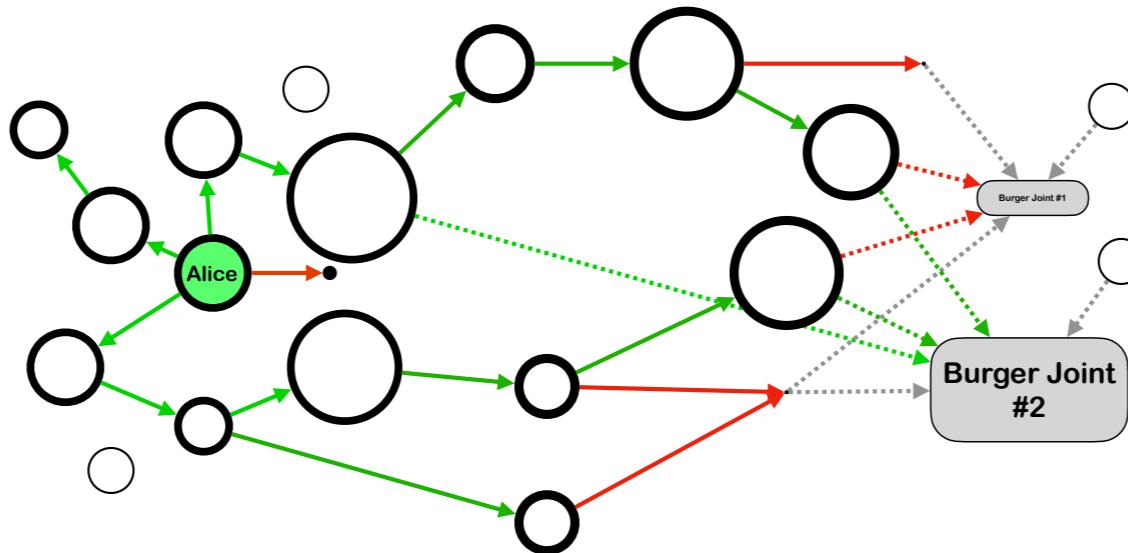
And you see that Burger Joint 2 is preferred over Burger Joint 1. Hooray! Better than Yelp! Right? Well, let's find out.
You decide, just for the fun of it, you'll shell out a few extra satoshis, and see what these other NON-trusted, probably SCAMbot users have to say.

Micropayments, monetization, and overturning today's Advertising-based business model



And guess what: your suspicions appear to be correct. All of their ratings are the opposite of what your trusty grapevine told you. Look at the two whose influence scores are practically zero; look at the two users about which you have absolutely zero information; they're all saying the opposite of what your trusted grapevine told you. It would seem they were misleading, perhaps they are employees of Burger Joint 1, or some state-controlled bots trying to make us miserable for only god knows why. Thank goodness you have your Grapevine not only to tell you who to trust — like a white list — but also to filter out the bad apples; a black list.

Micropayments, monetization, and overturning today's Advertising-based business model



You go to burger joint 2. You have a great time. You leave your own rating attestation. You've build up a good reputation. And a week later: you discover that people have paid you microfees for your attestation; you've made a couple bucks! Maybe not a lot, but kinda FUN! You decide from now on, whenever you have information that you think someone else may find useful, you'll encode it as an attestation, maybe make a few more bucks. And you know that if you leave crappy information — a shitty review that you're doing out of spite, or bc you're having a bad day, not one that's truly informative — you run the risk that your influence score will suffer, and you will make less money.

Micropayments, monetization, and overturning today's Advertising-based business model

- better information for you
- incentivizes you to input USEFUL information
- DESTROYS the legacy Big Tech ad-based, soul crushing business model

Not only does this system proved you with better, more useful, more accurate, information.

It also turns monetization on its head. You are monetizing your data. Without advertising! Ultimately that monetization will find its way into the hands of developers, who will provide you with better and better tools to make this system run. You want to make money? All you have to be is honest, and observant. It probably won't make you rich. But making people rich is not what this is for so that's OK. Destroying Big Tech soul-crushing advertising model; that's what I want to see happen.

[end 24]

25

The GRAPEVINE App

[start 25]

Returning now to the Grapevine app ...

The GRAPEVINE App

a Web of Trust

The Grapevine is essentially a Web of Trust.

The GRAPEVINE App

a Web of Trust Influence

Although Web of Influence might be a better name for it. Unlike all the WoT that have come before it, the Grapevine was designed specifically to pair up with the Concept Graph. And that has shaped its function and its design. Influence plays the role of WEIGHT: so that if an average score is calculated, instead of one user account = one vote, you give different weights to different user accounts, with weight being equal to the Influence Score.

The GRAPEVINE App

a Web of Trust Influence
Influence ~ Average Score

How exactly Influence is calculated is beyond the scope of this video. But suffice it to say that Influence is proportional to average trust score.

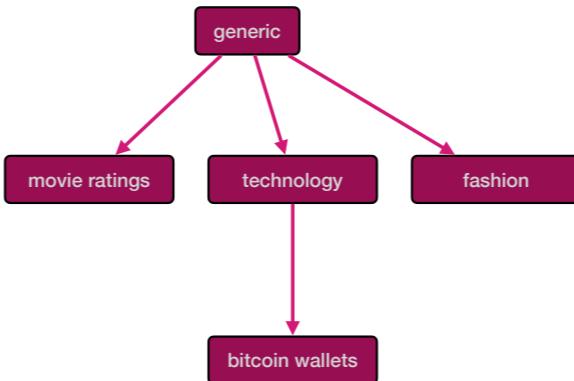
The GRAPEVINE App

a Web of Trust Influence
Influence ~ Average Score [context-based]

This score is context based.

The GRAPEVINE App

a Web of Trust Influence
Influence ~ Average Score [context-based]



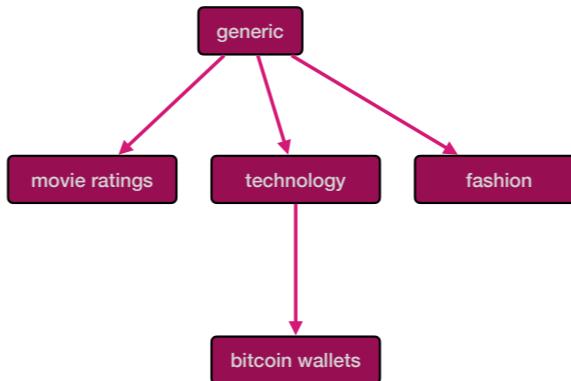
And context is hierarchical. Depicted here is a hierarchy of contexts, with generic on top; movie ratings, technology, etc. lower down on the hierarchy. I also refer to the hierarchy as the Context Tree. You may wonder: where does the context tree come from? Well, this is something you could manage yourself if you are so inclined; or you can allow your Grapevine to manage for you, or a combination of both. Most users will probably want the Grapevine to manage the bulk of the context tree; and then some users may make small edits here and there, on topics that are of particularly great interest to them. In this fashion, the context tree will be established and maintained in a manner that will let them take advantage of Loose Consensus for the Context Tree.

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

established via the
Loose Consensus of:



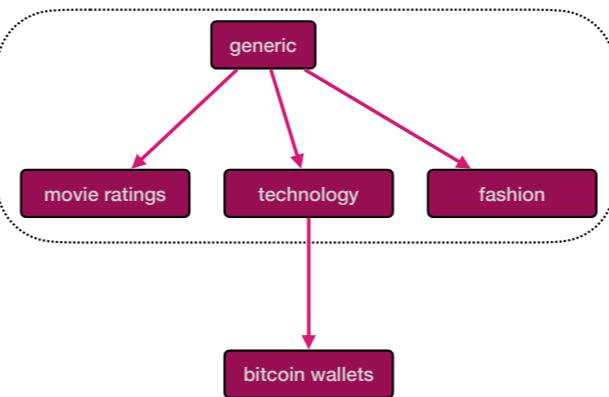
And what I mean by that is that the majority of the context tree could be selected by your main grapevine.

The GRAPEVINE App

a Web of Trust Influence
Influence ~ Average Score [context-based]

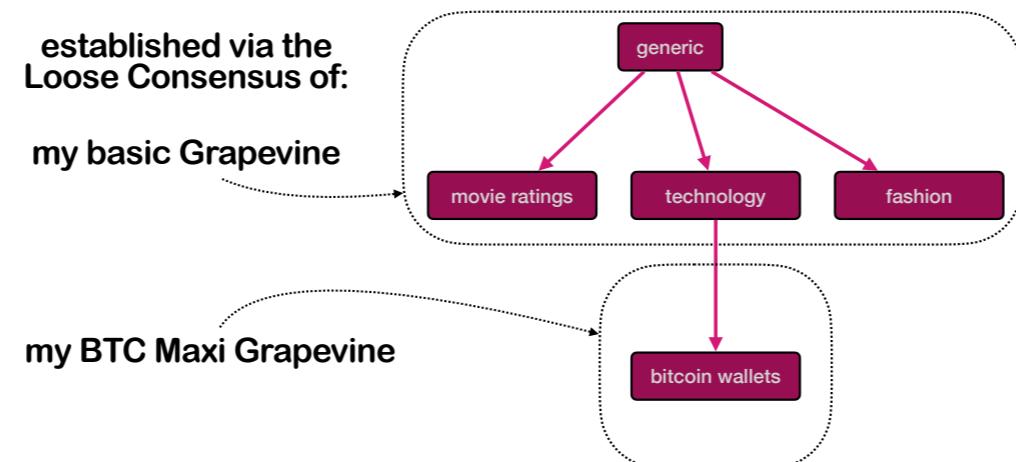
established via the
Loose Consensus of:

my basic Grapevine



The GRAPEVINE App

a Web of Trust Influence
Influence ~ Average Score [context-based]

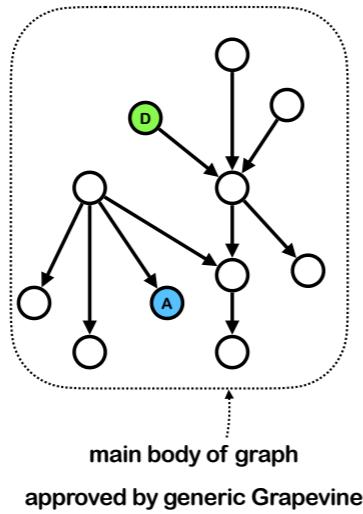


And then you could hand select specialized contextual subsets of your grapevine to superimpose small branches to your main context tree, to flesh out contexts that may not be in general usage.

Superpositioning of graphs

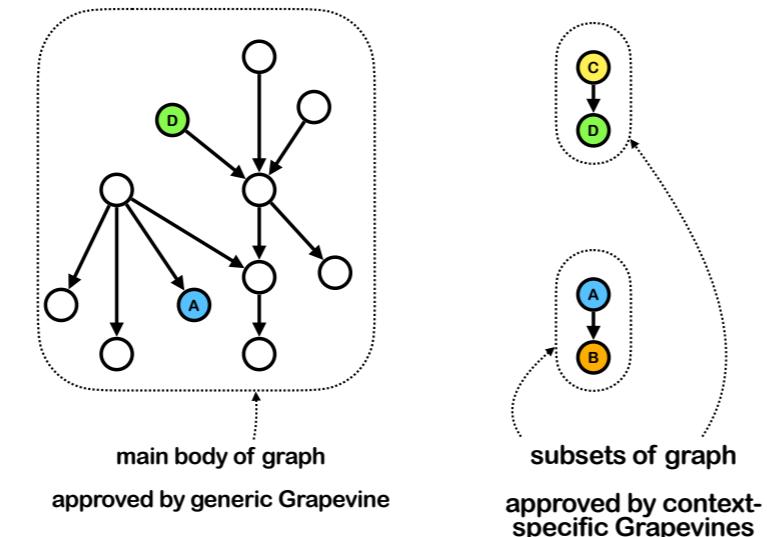
Loose Consensus of the Context Tree serves to illustrate why it is that I make such extensive use of graphs in the design of the Concept Graph and the Grapevine. Graphs can be superimposed (or superpositioned) by taking the union of two or more graphs to form another graph.

Superpositioning of graphs



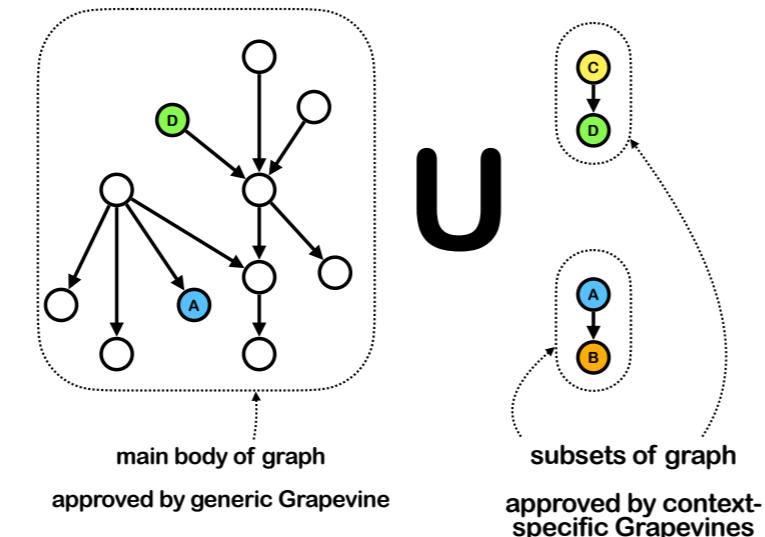
Suppose I have a graph that represents the Context Tree; or a Concept Graph, with a complex tree of sets and subsets; or a JSON Properties Tree. Suppose I use my Grapevine to approve the main body of the graph.

Superpositioning of graphs



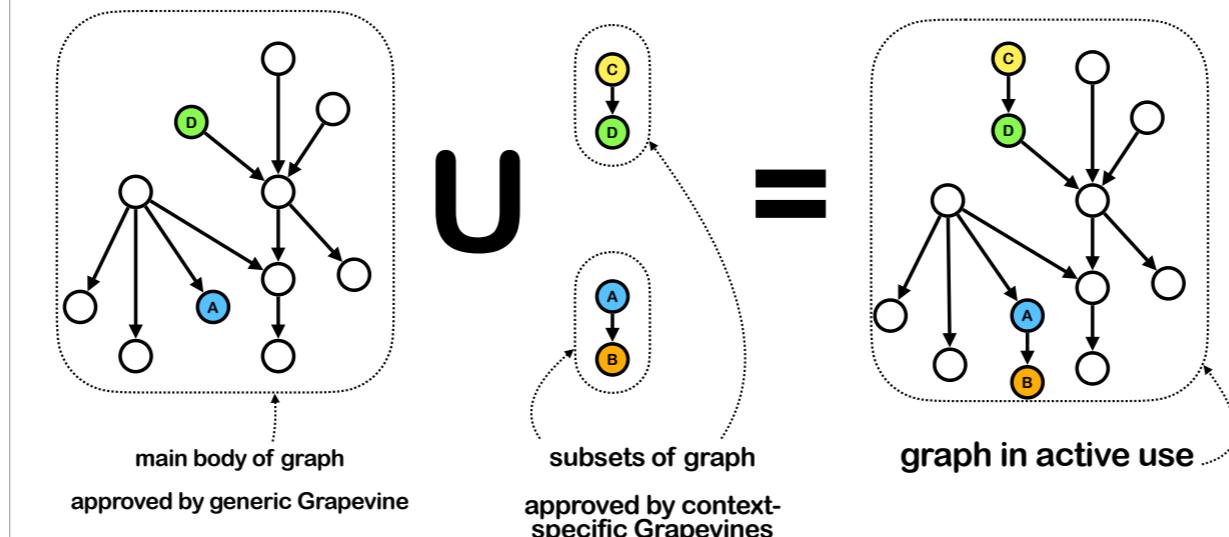
But then I have have small additions to the graph that have been approved by context-specific Grapevines.

Superpositioning of graphs



I can take the union of all of these graphs,

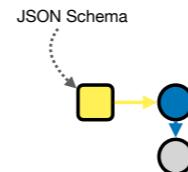
Superpositioning of graphs



and use that to form a new graph, shown here on the right, which is the one that I employ for active use in my app.

Superpositioning of graphs

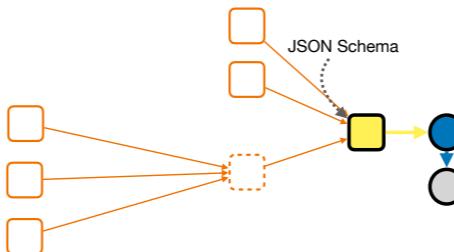
The ability to superposition multiple graphs together is why it makes sense to represent JSON Schema as a graph (which is what a property tree is for).



The ability to superposition multiple graphs together is why it makes sense to take a JSON Schema, shown here in yellow,

Superpositioning of graphs

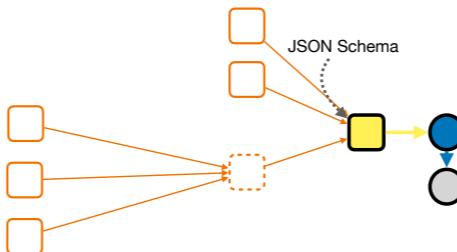
The ability to superposition multiple graphs together is why it makes sense to represent JSON Schema as a graph (which is what a property tree is for).



and break it apart into a graph of individual Properties, as I did earlier in this presentation.

Superpositioning of graphs

The ability to superposition multiple graphs together is why it makes sense to represent JSON Schema as a graph (which is what a property tree is for).

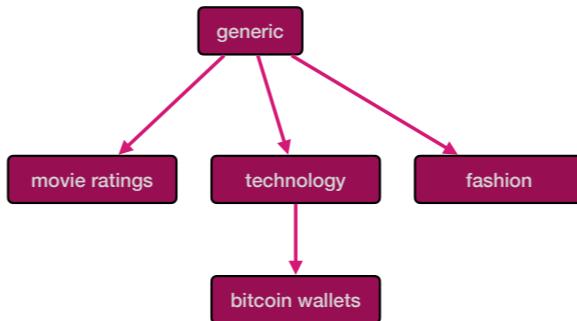


- **JSON Properties tree**
- **Context tree**
- **Concept Graph (Loki Pathway layer)**

So the JSON Properties Tree

Superpositioning of graphs

The ability to superposition multiple graphs together is why it makes sense to represent JSON Schema as a graph (which is what a property tree is for).

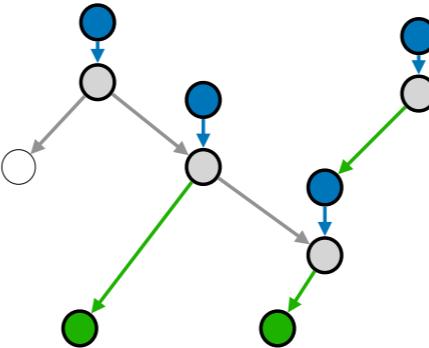


- JSON Properties tree
- Context tree
- Concept Graph (Loki Pathway layer)

the Context tree

Superpositioning of graphs

The ability to superposition multiple graphs together is why it makes sense to represent JSON Schema as a graph (which is what a property tree is for).



- **JSON Properties tree**
- **Context tree**
- **Concept Graph (Loki Pathway layer)**

and the Concept Graph (which is the graph that has Loki Pathways)
all make use of Superpositioning as Loose Consensus is established.

[end 25]

26

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

[start 26]

Influence is also proportional to confidence in the average trust score. Confidence is a number between 0 and 1. The average score is calculated from multiple individual ratings, trusted ones that is, and as the number of input ratings increases, the confidence asymptotically approaches 100 percent, i.e., 1.

[end 25 – depr]

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Kim Kardashian
1M followers

Joe Schmo
1K followers

[start 26 — depr]

You might ask: why do it this way? Answer: I consider it VERY IMPORTANT TO AVOID calculating influence the way current existing proprietary web 2.0 platforms calculate influence. For example: On Instagram, compare a user with 1K followers with a user who has 1M followers. Following someone is kinda sorta treated like a trust rating, although of course it is problematic in so many ways: it's not contextual; and it's not really trust that it's measuring, it's attention. Guess what?

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

KIM
KARDASHIAN

Joe Schmo

the user with 1M followers gets 1000x more influence than Joe Schmo with 1K followers.

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Kim Kardashian
1M ratings

Joe Schmo
1K ratings

Well, that's not the way the grapevine works. In the Grapevine, the user with 1M — I'm going to say 1M rating attestations rather than 1M followers, bc that's how the grapevine works — will have MARGINALLY more influence than the one with 1K ratings.

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Kim Kardashian
1M ratings

0.999

Joe Schmo
1K ratings

0.94

Why is that? because the CONFIDENCE in whichever average trust score we're looking at, is already pretty high with 1K ratings. Here, it is 0.94, hypothetically speaking. It's close to unity. Going from 1K to 1M ratings doesn't raise your confidence in the average trust score all that much. From 0.94 to 0.999; influence score isn't affected that much.

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Kim Kardashian
1M ratings

0.999

Joe Schmo
1K ratings

0.94

Howdy Doodie
1 rating

0.2

Now, if we were to encounter a user with ONE rating only, then sure, the confidence would be pretty low. 0.2 in this hypothetical calculation

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Kim Kardashian
1M ratings

0.999

Joe Schmo
1K ratings

0.94

Howdy Doodie
1 rating

0.2

This illustrates another reason why the advertising model must be destroyed. It rewards attention, bc attention means more eyes, more advertising revenue. But the Grapevine does not serve that master. That's not what the Grapevine is designed to calculate. It is designed to estimate how much influence that YOU want someone to have in YOUR grapevine. And that is simply a different metric. In the grapevine, advertisers are not the master; YOU are the master. That is because monetization doesn't accrue to advertisers; it accrues to YOU.

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Last thing to say about calculation of the Influence score is to mention how default scores work. The primary inputs to influence in any given context will be ratings that are relevant to that context. But what if we don't have any relevant ratings yet? Say, because the grapevine doesn't yet have many users?

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Influence is “inherited” from higher up in the context tree

Well the next best thing is to inherit scores from higher up in the context tree. What if we don't have those either?

The GRAPEVINE App

a Web of Trust Influence

Influence ~ Average Score [context-based]

Influence ~ confidence in the average score [0,1]

Influence is “inherited” from higher up in the context tree

Default scores for completely unknown users

Well the next best thing is to apply a default score for users about whom we have essentially ZERO information. Alice may want to set the default influence score to some average number for unknown users; but if sybil attacks are of concern to her, and they should be, she may choose to set the default score so some very small value, perhaps to zero. And the great thing is that she can always change this default setting. The defaults that she uses don’t have to be what someone else users. I’ll be showing examples of how all that works in other parts of this video.

[end 26]

27

The GRAPEVINE App

Front end:

- leave ratings

[start 27]

OK so, how does the front end of the Grapevine app work? just like the CG, the front end is designed to be user friendly. One of the things that the user will be able to do on the front end will be to leave rating attestations of other users, or posts, or listings, etc; although some of that will be expected to take place on other more specialized apps, like the decentralized Twitter app, for example.

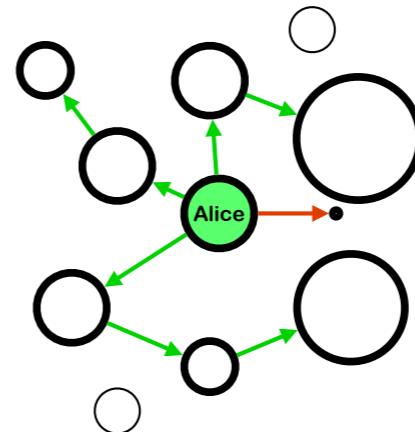
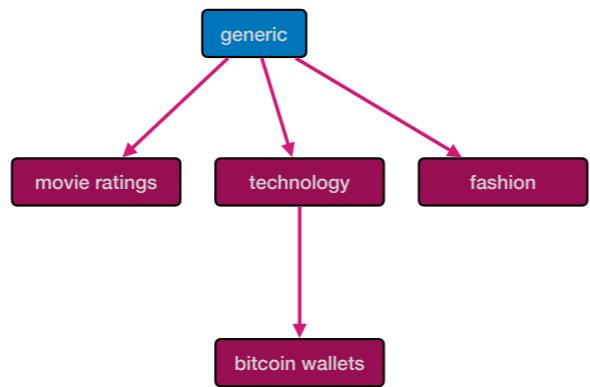
The GRAPEVINE App

Front end:

- leave ratings
- view Grapevine, with Influence scores

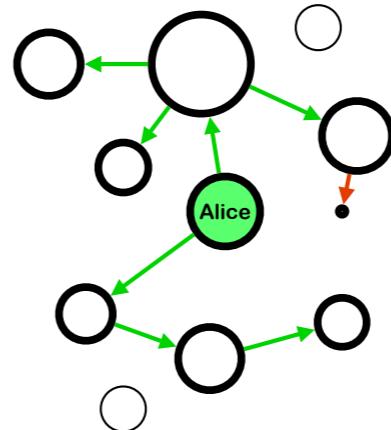
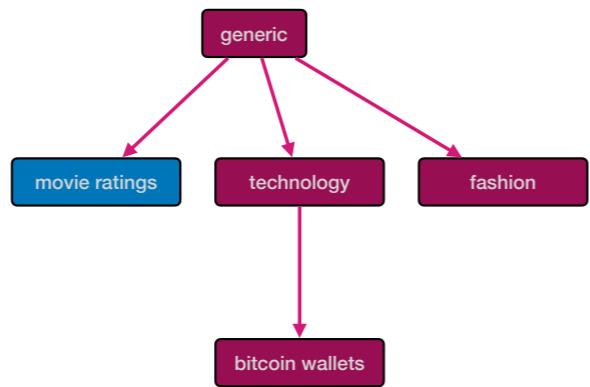
More importantly, on the front end, the user can explore the Grapevine and look at the influence scores of other users, in a context specific manner.

The GRAPEVINE App



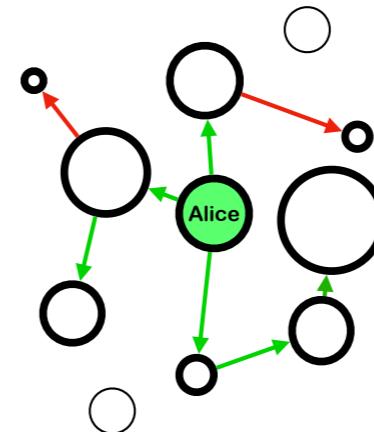
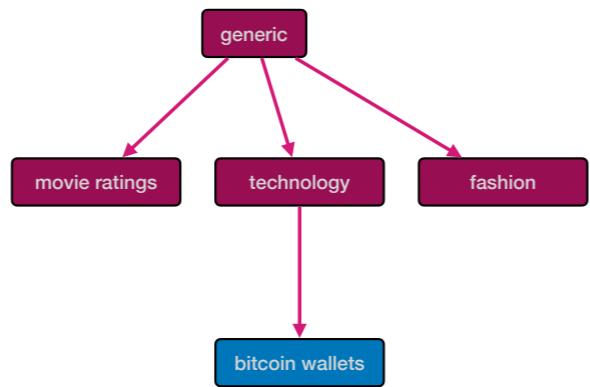
For example: this would be a view of Generic Influence scores of Alice's Grapevine, along with relevant ratings that are involved in calculating those scores.

The GRAPEVINE App



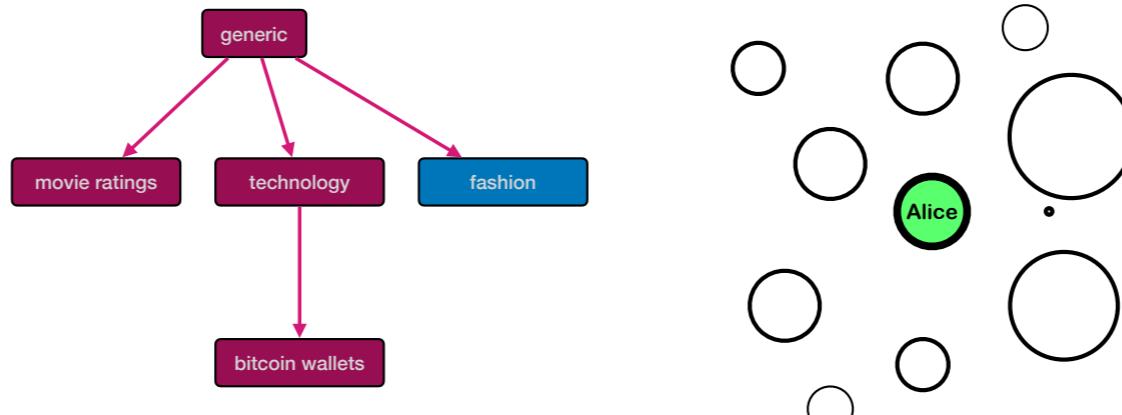
Likewise for Influence scores in the context of movies.

The GRAPEVINE App



Or bitcoin wallets.

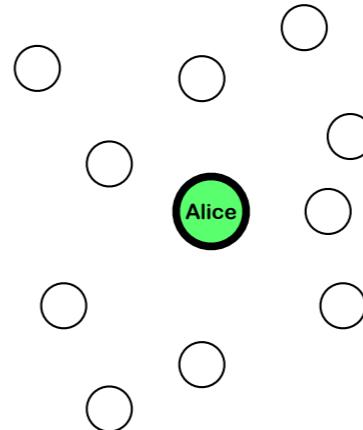
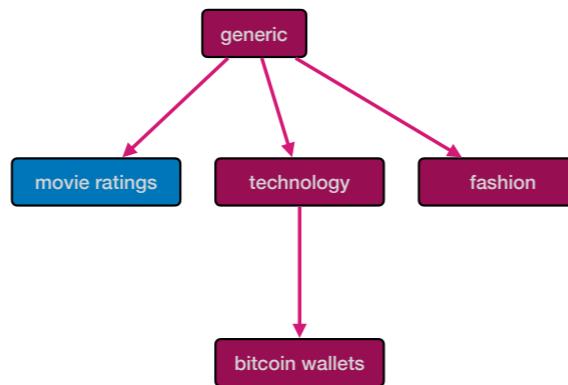
The GRAPEVINE App



Or fashion. In this case, there aren't any arrows between users on the right, because there aren't any ratings to show that are considered to be relevant to this context, bc this context is a recent addition to the context tree. So, no one has left any ratings yet. Therefore the scores you see on the right are inherited from higher up in the tree. As time goes on, ratings that are relevant to fashion, and are selected to be inputs for this influence score will be added. The information available to Alice through her Grapevine will get incrementally, but over time substantially, more rich over time.

Influence Context Tree

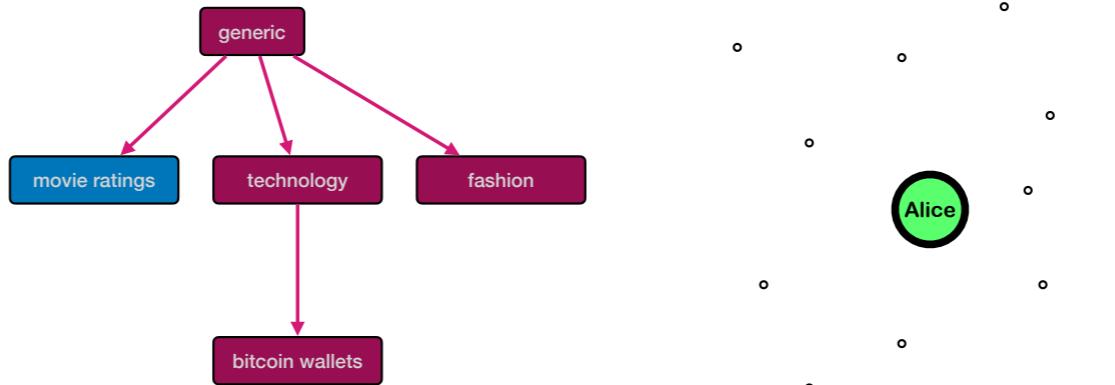
Step 1: set defaults for unknown users



So, to review how Influence Scores are calculated for a Context such as, let's say, movies. First, Alice sets the default scores for unknown users. The score might be intermediate, as you see here;

Influence Context Tree

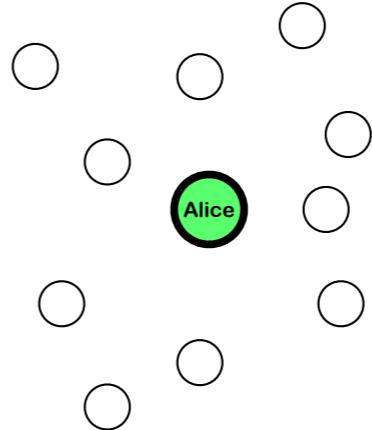
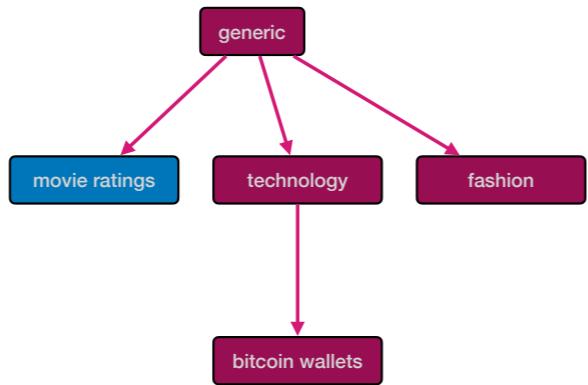
Step 1: set defaults for unknown users



or she might want to set it to very small, or maybe even zero, as you see here.

Influence Context Tree

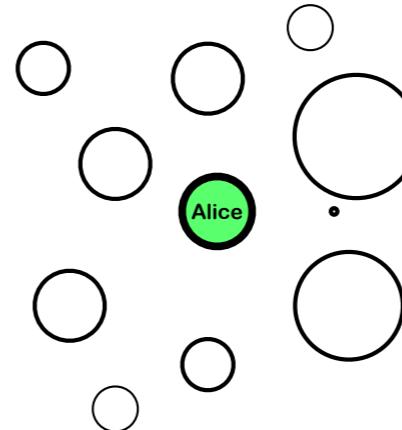
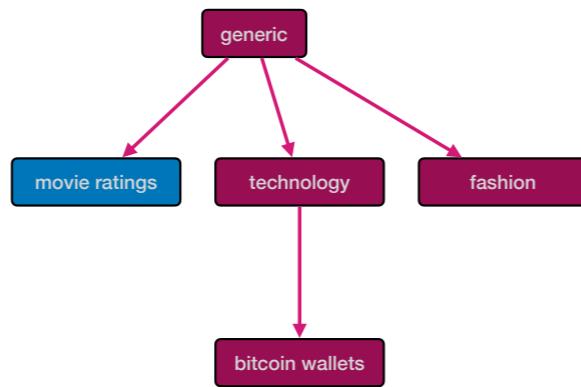
Step 1: set defaults for unknown users



Let's say she sets it to intermediate.

Influence Context Tree

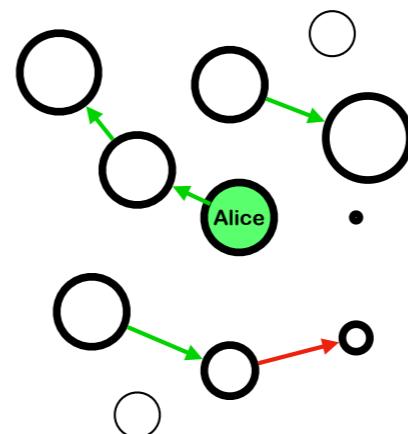
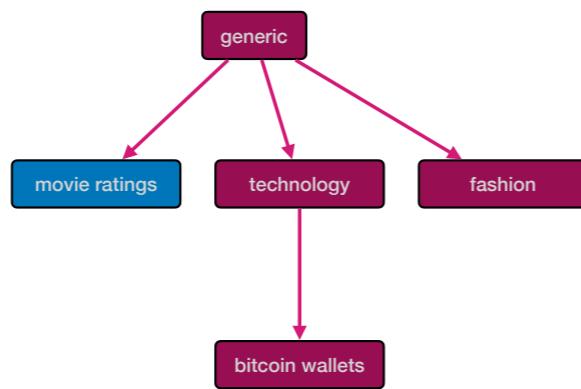
Step 2: inherit score from higher in the hierarchy



Next step is to inherit scores from higher up in the hierarchy of the Context Tree. In this case, it means inheriting scores from Generic Influence Scores.

Influence Context Tree

Step 3: input from relevant ratings



Finally, any user-to-user ratings that are relevant to movies are incorporated into the calculation. Alice might pick by hand which types of user-to-user ratings should be used in this calculation, or she might let her Grapevine pick for her, depending on how much time she wants to spend fine-tuning her Grapevine.

The GRAPEVINE App

Front end:

- leave ratings
- view Grapevine, with Influence scores

So that's how Alice could view her Grapevine and view and manage calculation of Influence Scores in a contextual manner.

The GRAPEVINE App

Front end:

- leave ratings
- view Grapevine, with Influence scores
- create new ratings and new ways to calculate average scores

The other thing Users will be able to do will be able to create new ratings and new ways to calculate average scores. As before, some of this might happen on more specialized apps, like an app for decentralized Twitter, where, for example, she might decide to create a rating, the purpose of which is to flag a user that is a troll or that is issuing spam. Conversely, she might want to create a rating the purpose of which is to identify users that post high quality content. These ratings can then be used to filter in or filter out content, using criteria that are under her control.

The GRAPEVINE App

Back end:

- communicate with other users (over IPFS, for example?)
- handle micro sales and micro purchases (interface with BTC/LN node?)
- calculate composite scores

OK so, how does the back end work? Well, a lot is already covered. But just to recap some of the functions:

It will manage communication with other users;

it will manage micro sales and micro purchases, as reviewed earlier in this presentation;

and it will calculate composite scores, as reviewed earlier in this presentation.

[end 27]

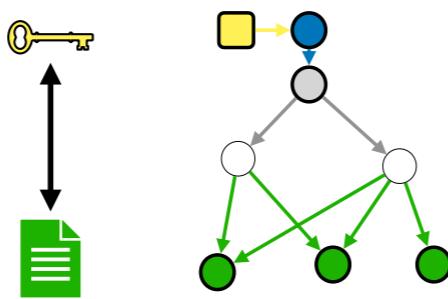
28

The Principle of Loki and the Cerebral Cortex

[start 28]

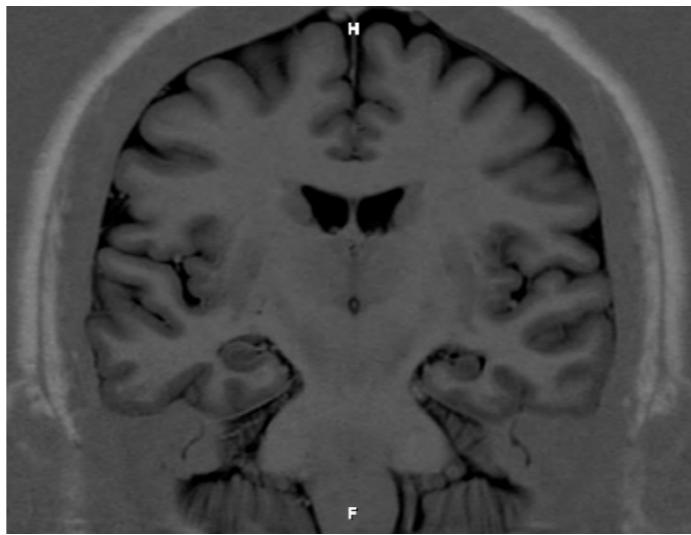
Let me turn attention now to the idea that the Principle of Loki already exists as a feature of the Cerebral Cortex.

The Principle of Loki and the Cerebral Cortex



The gist of the analogy is as follows. Recall that each node in the concept graph corresponds to a data file. And each data file has a format.

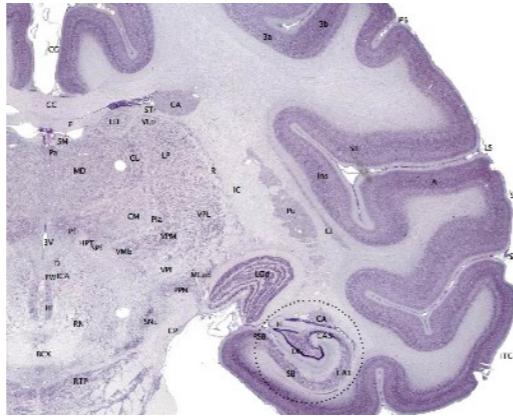
The Principle of Loki and the Cerebral Cortex



<https://www.EEGAtlas-online.com>

Turning our attention to the brain - Here is an MRI of the brain, taken from my EEG website that I threw together a couple years ago.

The Principle of Loki and the Cerebral Cortex

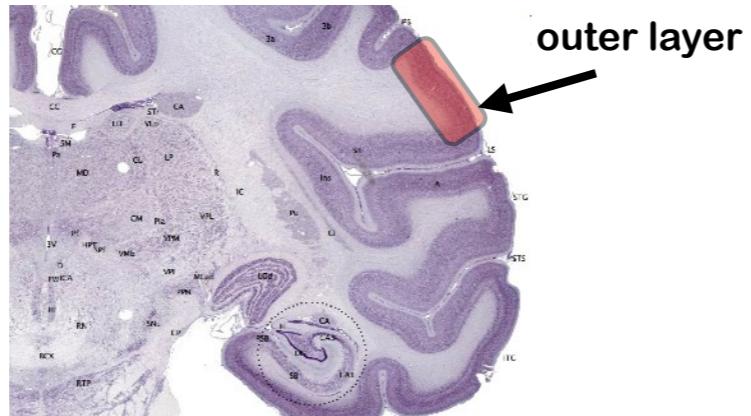


https://en.wikipedia.org/wiki/Cerebral_cortex

and here is a tissue slice from the brain of an adult macaque monkey that I pulled from wikipedia.

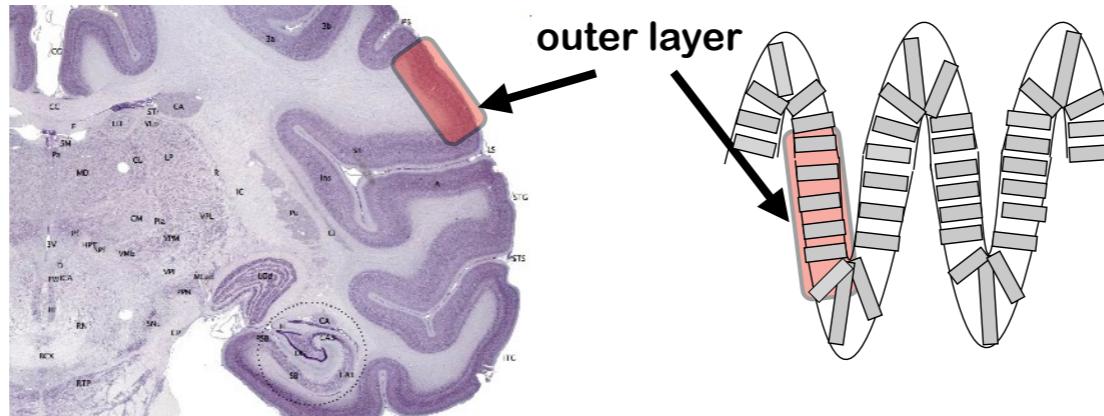
The cortex is the outer layer of neural tissue that covers the cerebrum of the brain in humans and other mammals. It is the darker blue layer in this picture. In humans it is roughly 2-3 mm in thickness.

The Principle of Loki and the Cerebral Cortex



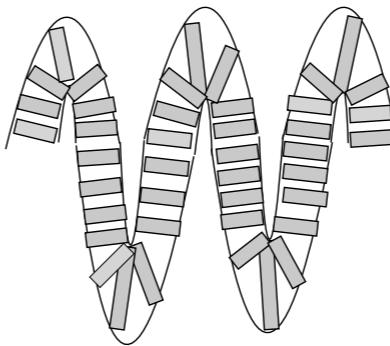
Here is one little segment of it. As you can see it has multiple folds and gyrations and covers most of the brain - basically the entire cerebrum. The nuclei of many of the neurons of the brain reside inside the cortex. The cortex is part of what we call grey matter. The lighter blue part underneath is called the white matter, and it contains axons, which function as projections from one neuron to another.

The Principle of Loki and the Cerebral Cortex



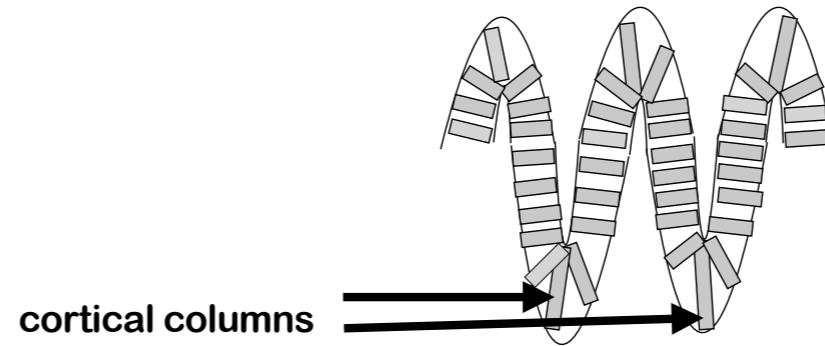
In this talk I will be depicting the cortex using a diagram like the one on the right, which illustrates several folds of the cortex, one after another

The Principle of Loki and the Cerebral Cortex



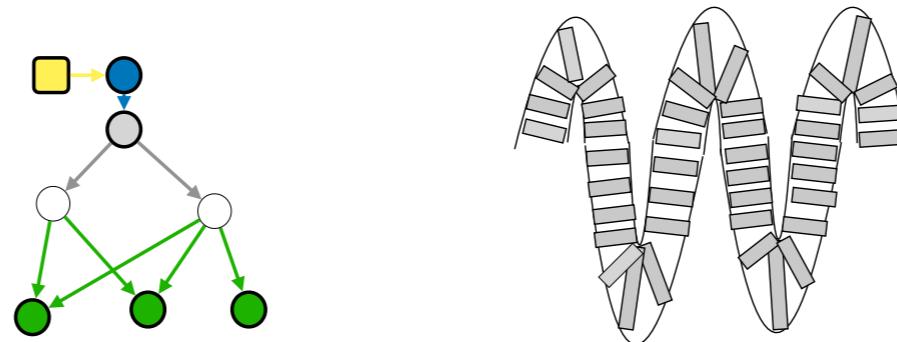
Inside the cortex, there are rectangles

The Principle of Loki and the Cerebral Cortex



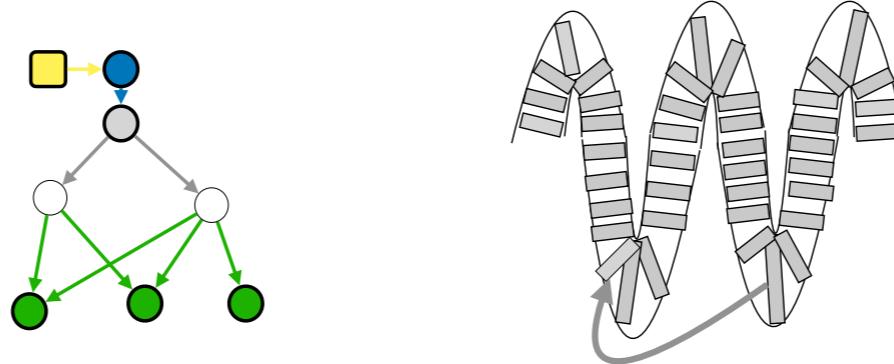
that I am using to represent what are known as cortical columns. They're not drawn to scale.
We're not sure what cortical columns do, exactly.

The Principle of Loki and the Cerebral Cortex



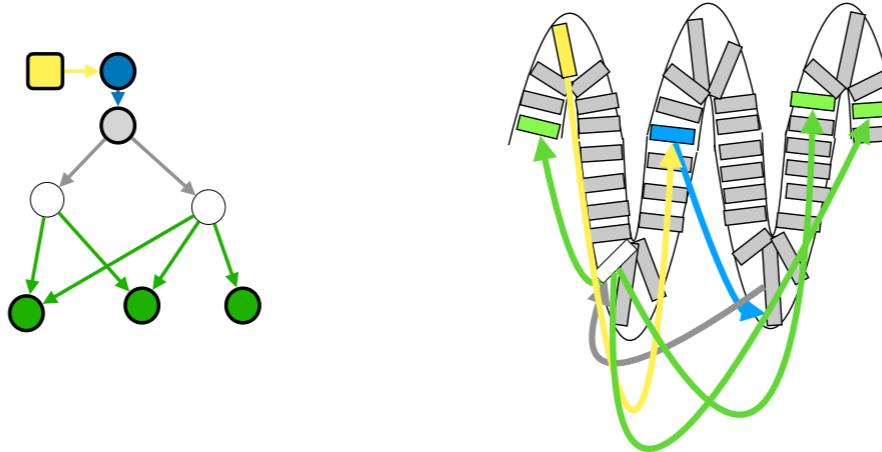
But let's imagine that each cortical column is like a data file. It stores information in it, somehow. And just like a data file, let's imagine that the cortical column has a format. And that there are an unlimited number of ways to format the data inside a cortical column. If that is true, then there must be a place somewhere in the brain, nearby, accessible, that has a record of the format of the column.

The Principle of Loki and the Cerebral Cortex



We already know that axonal projections exist that connect one cortical column to another. Here's an arrow that illustrates one of those projections, running through the grey matter.

The Principle of Loki and the Cerebral Cortex



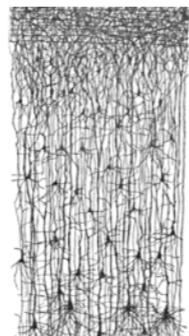
So all we have to do is imagine that the principle of Loki, as applied to data files in the concept graph on the left, applies in essentially the same way to cortical columns in the cortex on the right. So on the right, you can see there is one column that is yellow, and three that are green; and this would mean that the format of the data inside the each of the three green cortical columns is recorded inside the yellow cortical column. Furthermore, we imagine that that there exist pathways from one column to another, actual physical pathways, composed of axonal projections, that form the basis of Loki Pathways through the cortex.

The Cerebral Cortex and the Decentralized Web

So let me explain this analogy in a little more detail.

The Cerebral Cortex and the Decentralized Web

billions of neurons

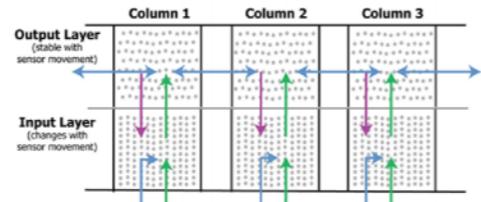


The cortex contains billions of neurons.

The Cerebral Cortex and the Decentralized Web

billions of neurons

millions of cortical columns



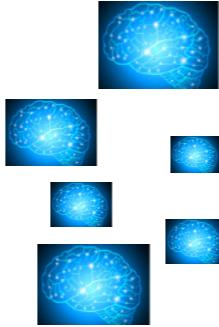
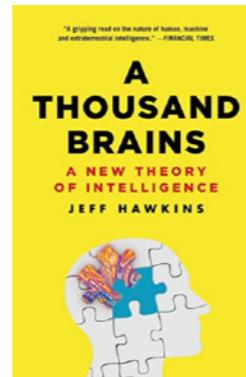
They are arranged into millions of cortical columns.

The Cerebral Cortex and the Decentralized Web

billions of neurons

millions of cortical columns

a Thousand Brains



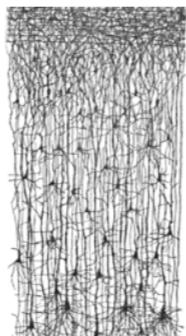
We still don't really know how knowledge is stored in the brain, or how the brain thinks. But one thing we know, is that there is a lot of parallel processing going on. And there is a model called the Thousand Brains model, described in this book, published in 2021 by Jeff Hawkins, that paints a picture according to which what we consider to be a single brain acting in unison is more like a huge number of brains (say, 1000), all existing in parallel.

The Cerebral Cortex and the Decentralized Web

billions of neurons

~

billions of users



The decentralized web has billions of users.

The Cerebral Cortex and the Decentralized Web

billions of neurons

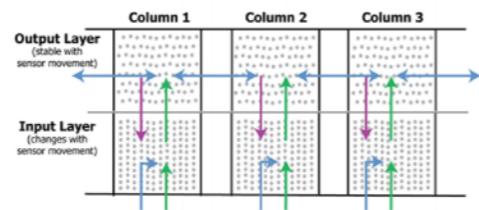
~

billions of users

millions of cortical columns

~

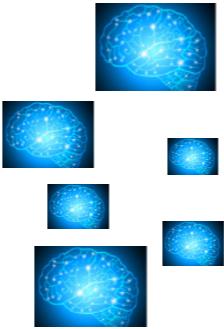
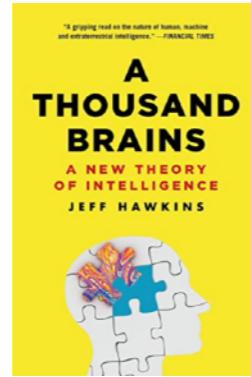
millions of developers



Millions of developers.

The Cerebral Cortex and the Decentralized Web

billions of neurons ~ billions of users
millions of cortical columns ~ millions of developers
a Thousand Brains ~ **a Thousand Platforms**



Reddit
IPFS
Blockstack
Twitter
Instagram
Haven
OpenBazaar

closed platforms; maintained by a tech company
open source platforms; maintained by a team of open source devs

And is composed of a huge number of platforms, all existing in parallel. Maybe not 1000, exactly, but, it's a lot.

The Cerebral Cortex and the Decentralized Web

billions of neurons ~ billions of users

millions of cortical columns ~ millions of developers

a Thousand Brains ~ a Thousand Platforms

both decentralized:

no neuron, column, or “brain” has authority over the others

In each case, the cortex and the web, they are decentralized.

No neuron, no column, no single one of the 1000 brains exercises authority over the others.

The Cerebral Cortex and the Decentralized Web

billions of neurons ~ billions of users

millions of cortical columns ~ millions of developers

a Thousand Brains ~ a Thousand Platforms

both decentralized:

no neuron, column, or “brain” has authority over the others no user, developer, or platform has authority over the others

Just like it is envisioned that no user, no developer, no platform in the decentralized web should exercise authority over the others.

The Cerebral Cortex and the Decentralized Web

billions of neurons ~ billions of users

millions of cortical columns ~ millions of developers

a Thousand Brains ~ a Thousand Platforms

both decentralized:

no neuron, column, or “brain” has authority over the others no user, developer, or platform has authority over the others

lack of consensus regarding DATA FORMAT

And so I am proposing that they both suffer from the same challenge: how to generate CONSENSUS regarding how to format data.

The Cerebral Cortex and the Decentralized Web

billions of neurons ~ billions of users

millions of cortical columns ~ millions of developers

a Thousand Brains ~ a Thousand Platforms

both decentralized:

no neuron, column, or “brain” has authority over the others no user, developer, or platform has authority over the others

lack of consensus regarding DATA FORMAT

WALLED GARDENS

They both therefore must overcome the challenge of Walled Gardens, which is what happens when all these disparate entities cannot or do not agree on how to format data.

Knowledge Representation in the Cerebral Cortex

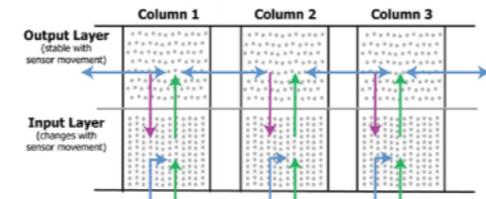
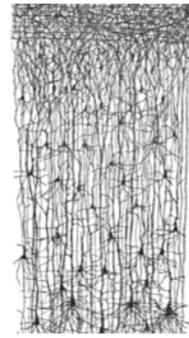
Columnar Hypothesis

Pattern Recognition Theory of Brain

Thousand Brains Hypothesis

I've stated that we don't know how knowledge is represented in the brain. But I'll review briefly a few models.

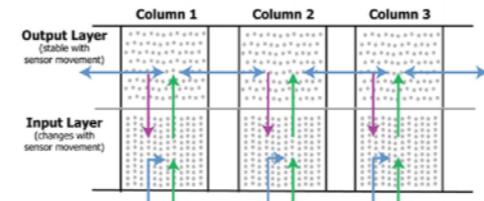
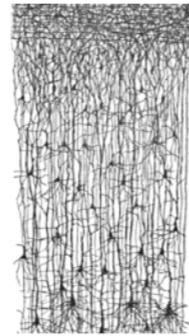
Mountcastle and cortical columns



<https://numenta.com/neuroscience-research/cortical-columns/>

In the middle of the 20th century, Mountcastle observed, under a microscope, that neurons in the cortex are arranged in structurally similar units, called cortical columns, that repeat all over the entire cortex.

Mountcastle and cortical columns

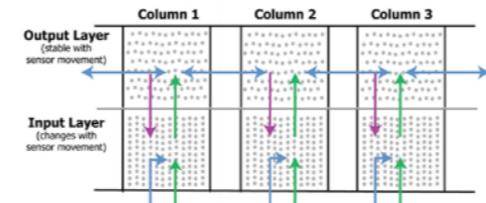
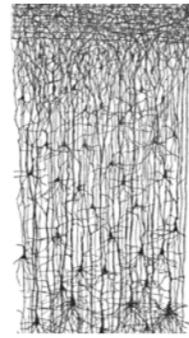


- $\sim 10^2$ neurons per minicolumn
- $\sim 10^2$ minicolumns per column
- $\sim 10^6$ columns in the brain

<https://numenata.com/neuroscience-research/cortical-columns/>

There are estimated to be between 10 to the 5th and 10 to the 6th cortical columns in the human brain.

Mountcastle and cortical columns



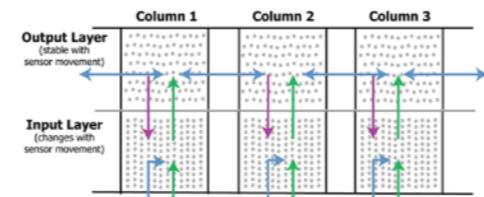
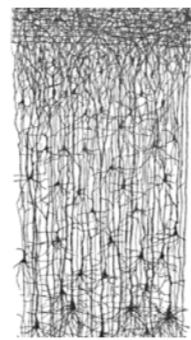
- ~100 neurons per minicolumn
- ~100 minicolumns per column
- ~1-2 million columns in the brain

The entire cortex is basically a bunch of cortical columns

<https://numenta.com/neuroscience-research/cortical-columns/>

so basically the entire surface layer of your brain is just a whole bunch of cortical columns.

Mountcastle and cortical columns

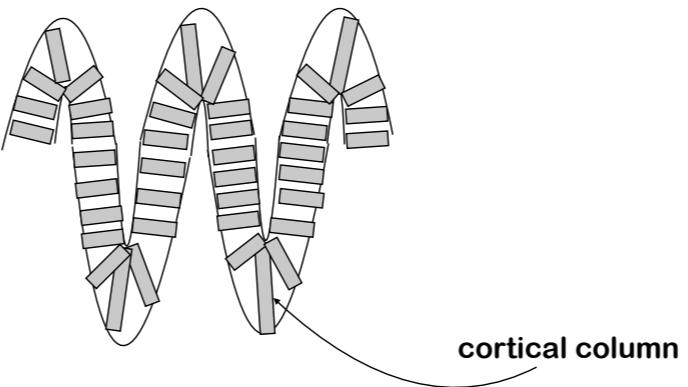


<https://numenta.com/neuroscience-research/cortical-columns/>

As of today, we still don't know the function of a column.

Columnar Hypothesis

cortical columns, which are repeated throughout the cortex, do something interesting (we just don't know what)

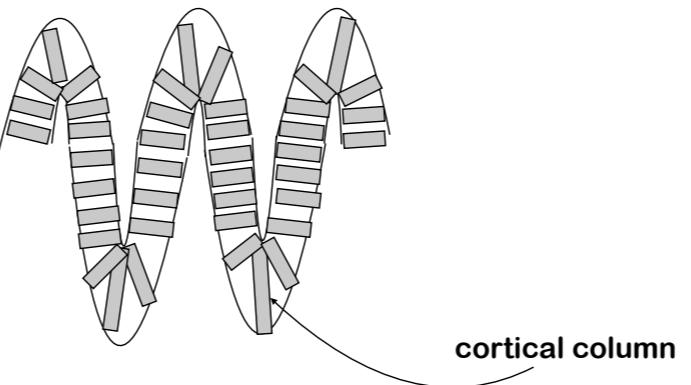


On the one hand, it could be that there really is no function. Perhaps it's just a convenient way to package neurons together, like eggs in a carton, and it's nothing more interesting than that.

Columnar Hypothesis

cortical columns, which are repeated throughout the cortex, do something interesting (we just don't know what)

each column = a functional unit

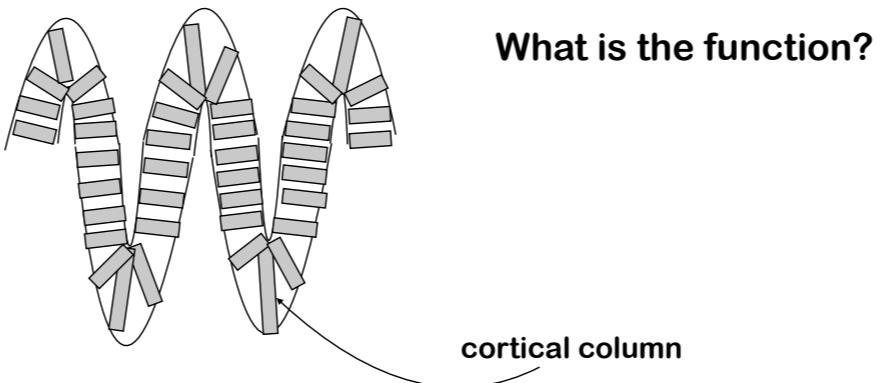


But most, I think, favor the idea that the column has some sort of function. IOW the column acts as some sort of functional unit. I'll refer to that as the columnar hypothesis.

Columnar Hypothesis

cortical columns, which are repeated throughout the cortex, do something interesting (we just don't know what)

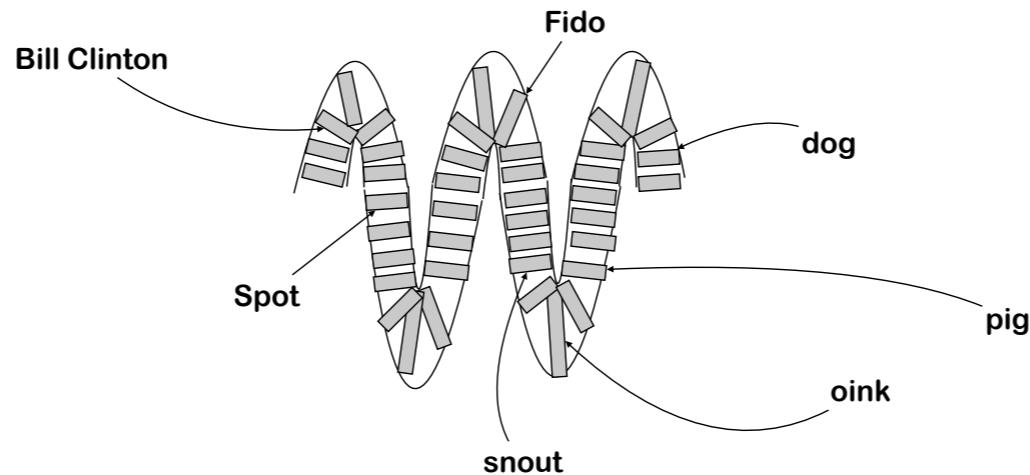
each column = a functional unit



If so, then: what is the function? Guess what - we don't know!

Pattern Recognition Theory of Brain

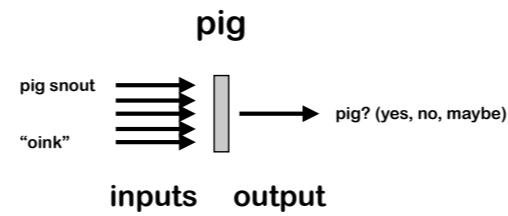
each cortical column recognizes a “pattern”



But we do have ideas! One idea is something called the Pattern Recognition Theory of Brain. According to the Pattern Recognition Theory of Brain, the basic function of a column is to recognize a particular pattern.

Pattern Recognition Theory of Brain

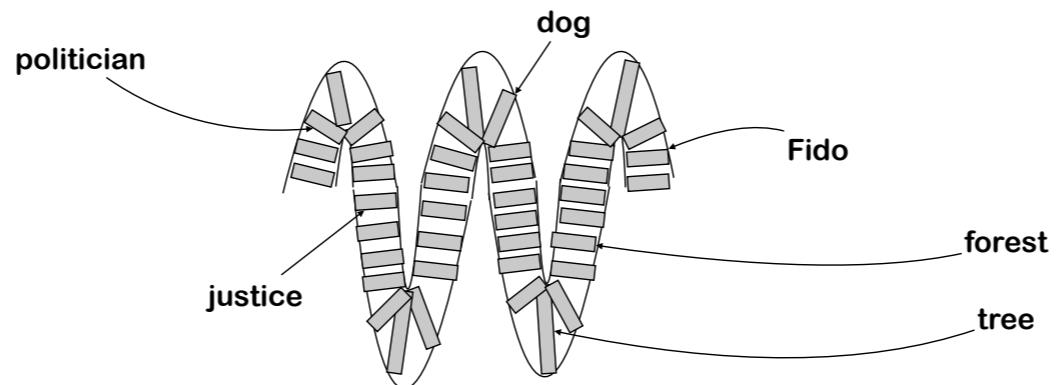
each cortical column recognizes a “pattern”



in other words: each column takes inputs, which may be from other columns, or could be one or two steps away from primary sensory data, like from our eyes. And then it outputs whether it sees the pattern. yes, no, maybe. That output becomes input for other columns. In this way, the cortical columns function in parallel, and yet there is also a hierarchical organization, with one set of columns providing inputs into another, higher in the hierarchy, so to speak.

Pattern Recognition Theory of Brain

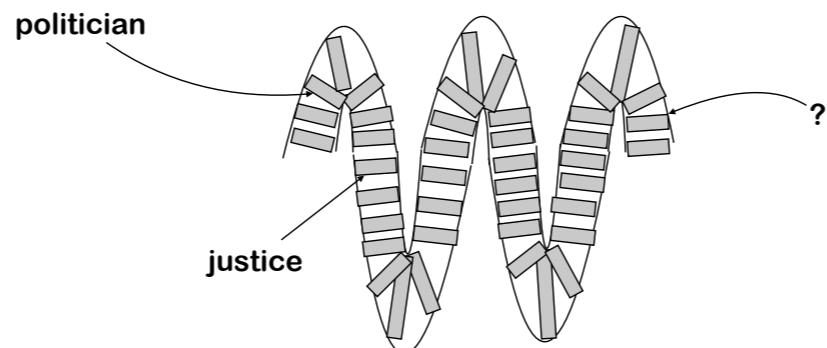
Patterns can be abstract or concrete; granular or coarse



Patterns can be anything. Abstract, concrete, granular, coarse.

Pattern Recognition Theory of Brain

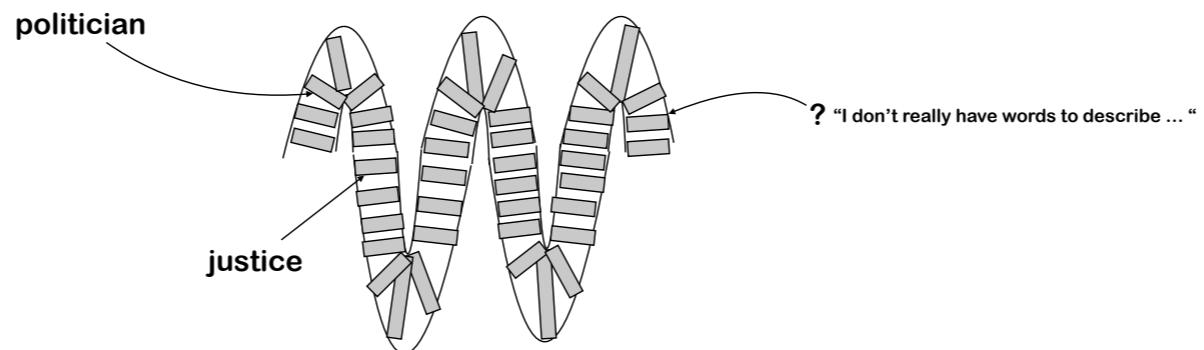
any given pattern may or may not map to a word or phrase



In many cases, a pattern will map to a word or a short phrase. But in some cases, a pattern may not be associated with a word or phrase.

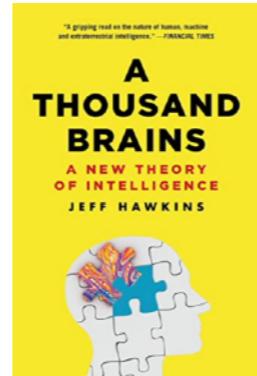
Pattern Recognition Theory of Brain

any given pattern may or may not map to a word or phrase



We all know what it feels like to have thought, that we can hold in our heads, for which we don't have a word or a label. Which can make it difficult to communicate, sometimes.

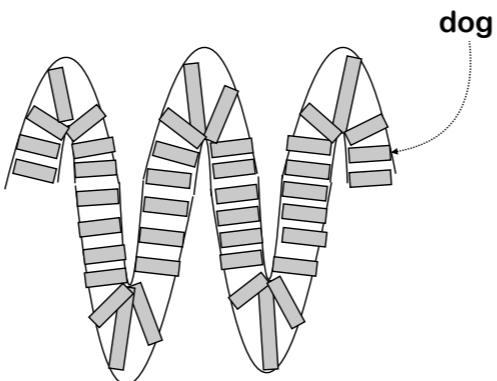
**Pattern Recognition Theory of Brain
augmented by
the Thousand Brains Hypothesis**



I mentioned earlier a book called the Thousand Brains, by Jeff Hawkins, published very recently, in 2021.

Thousand Brains hypothesis

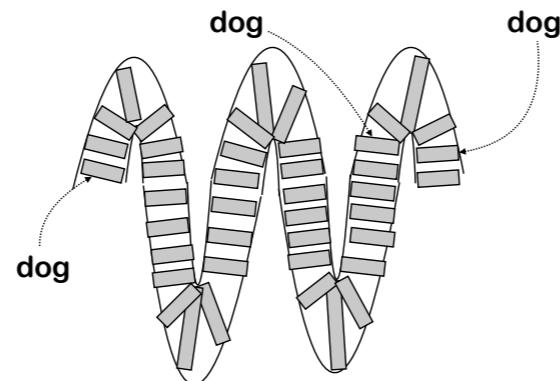
each “pattern” is represented **many times**, in parallel



As I understand it, with apologies to Dr Hawkins in case I am saying this wrong: one way to explain the Thousand Brains model would be to observe that for any given pattern, like dog for instance, we don't just have one column to represent that pattern; we have lots.

Thousand Brains hypothesis

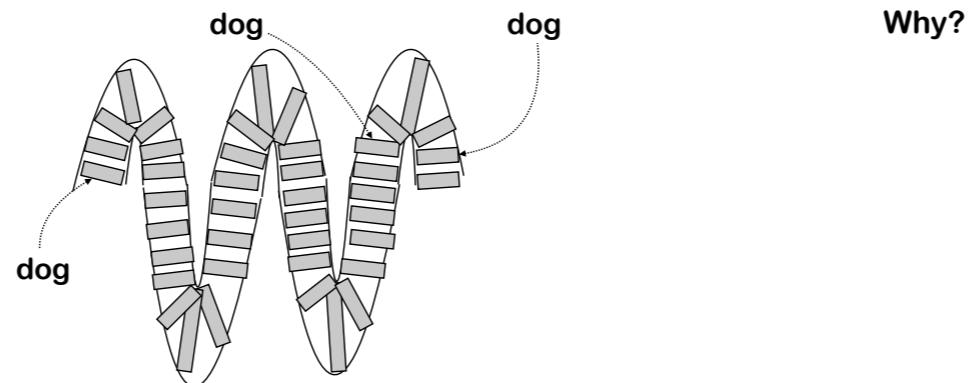
each “pattern” is represented **many times**, in parallel



I'll show 3 here, but imagine that there are lots, maybe 1000 columns that each represent basically the same pattern: dog.

Thousand Brains hypothesis

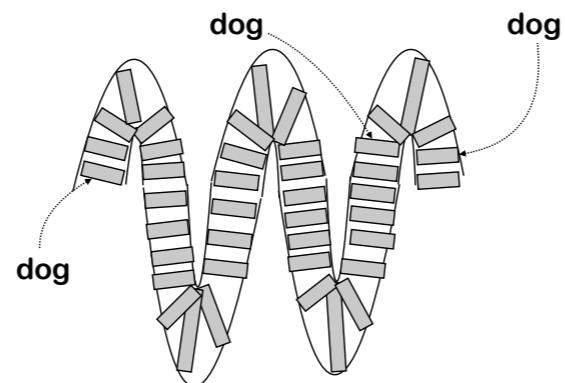
each “pattern” is represented **many times**, in parallel



You might wonder: why do this?

Thousand Brains hypothesis

each “pattern” is represented **many times**, in parallel



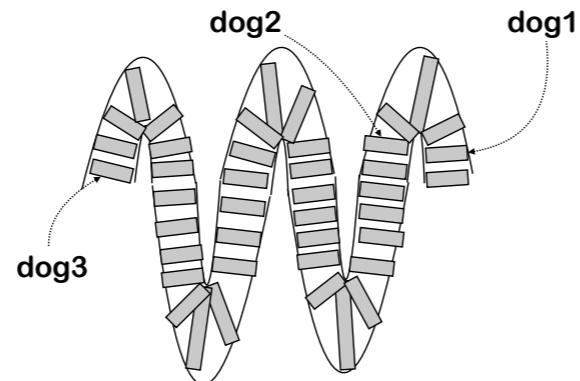
Why?

- redundancy

One reason might be simple redundancy: in case one fails, you have backups.

Thousand Brains hypothesis

each “pattern” is represented **many times**, in parallel



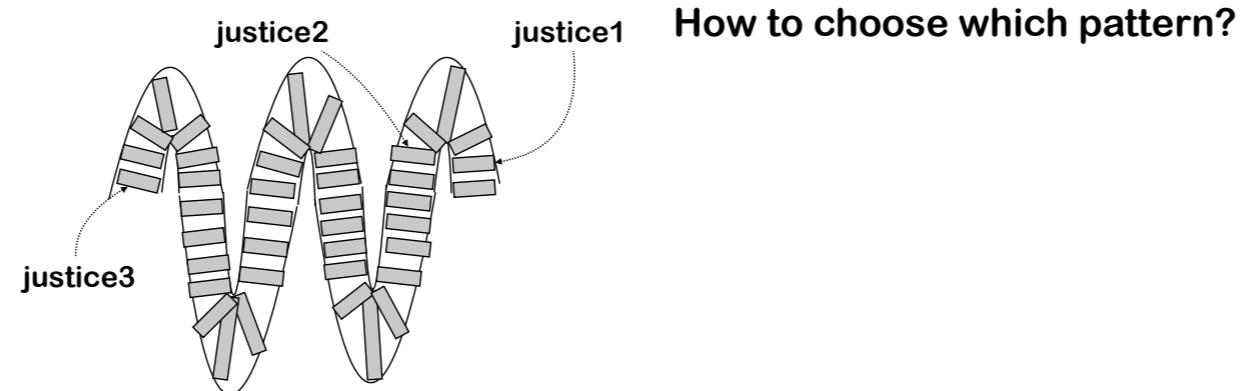
Why?

- redundancy
- variability

Another, slightly more interesting reason I think, would be variability. Perhaps each of the 1000 pattern recognizer columns is exactly the same. More realistically, the brain is a wet and messy place, they probably differ at least in small details. Even more interestingly: perhaps they differ in more important ways; maybe even conflicting ways. Consider an abstract idea, like justice: most of us probably have more than one way to define what that means; more than one way to define what we would be looking for if we were tasked with detecting its presence or its absence in, let's say, in a story.

Thousand Brains hypothesis

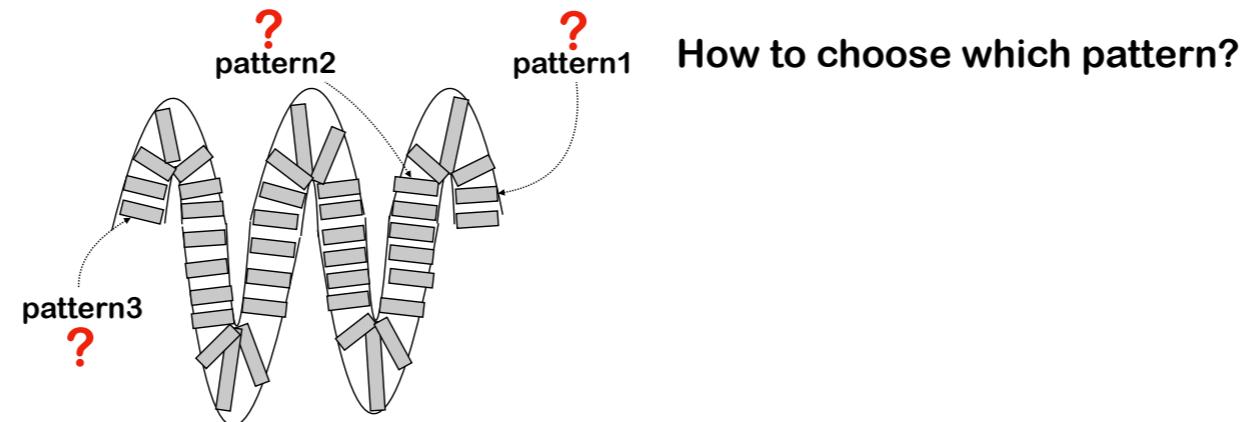
each “pattern” is represented **many times**, in parallel



A question that I do not think was raised in the book (or if it was, I missed it): if there are multiple patterns, and they are not exactly identical, then:

Thousand Brains hypothesis

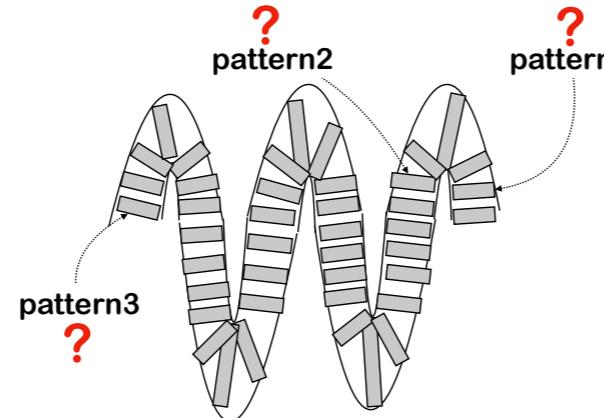
each “pattern” is represented **many times**, in parallel



which one would we use, at any given moment in time, or in any given context? How do we choose one pattern recognizer, meaning one cortical column, over the others?

Thousand Brains hypothesis

each “pattern” is represented **many times**, in parallel

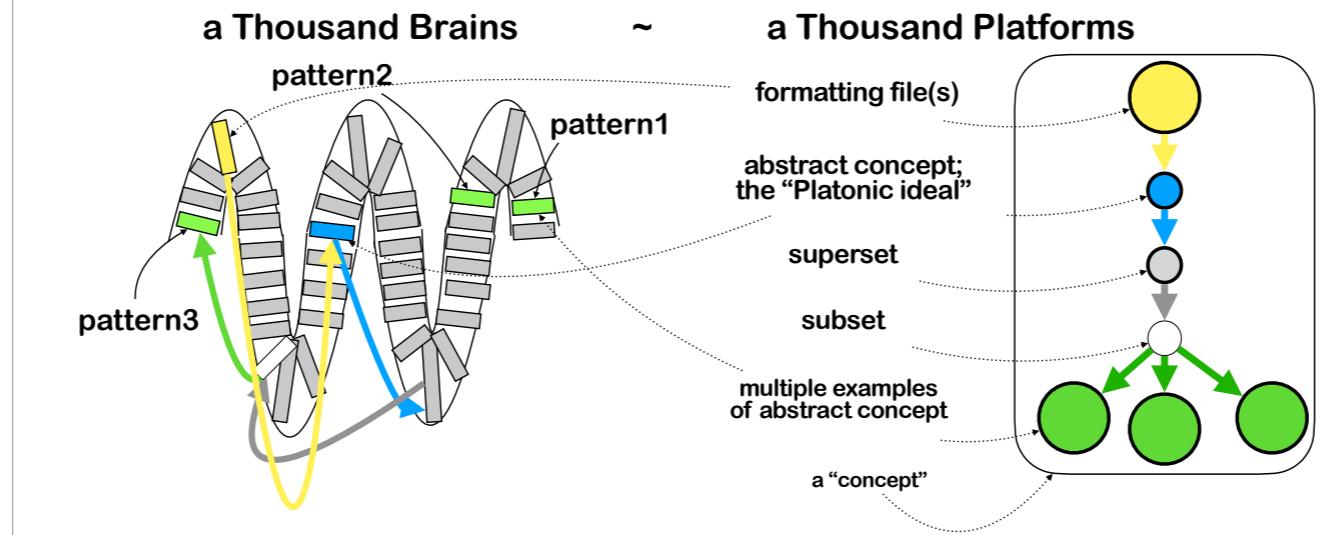


How to choose which pattern?

The Thousand Brains
model does not provide
an answer.

By my understanding, the Thousand Brains model does not provide an answer.

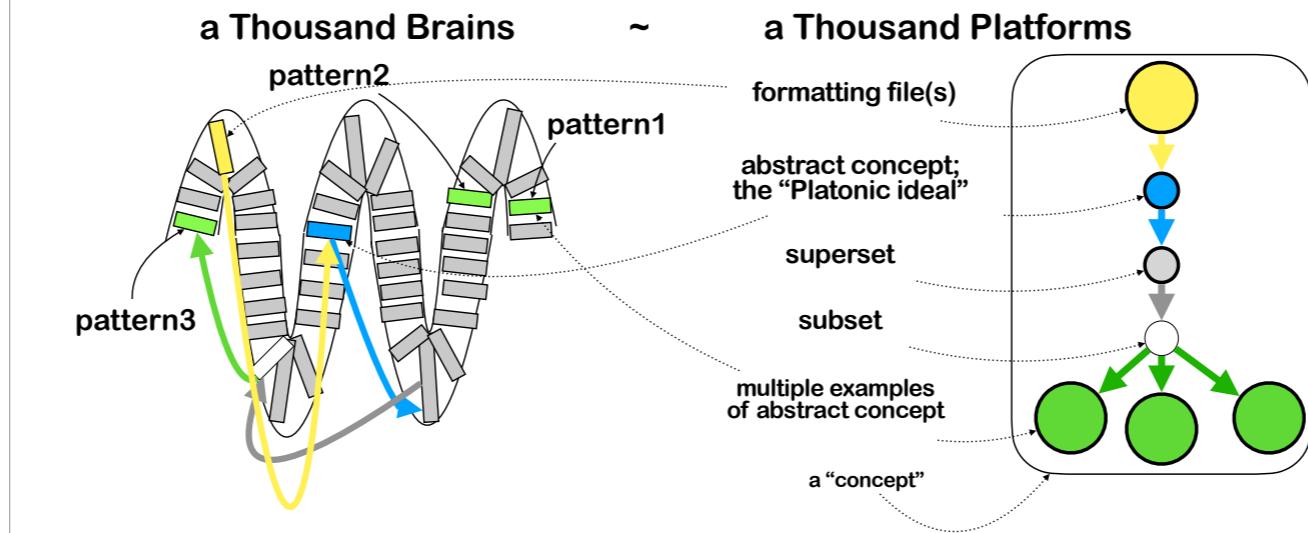
**Hypothesis: the optimal structure of the decentralized web
and the actual structure of the cortex follow the **Loki Principle****



And so, I would like to propose an answer: which is that the Loki Principle, as I have been describing it in this talk, applies to the cortex, in the same way that I have been arguing that it ought to be applied to the decentralized web.

Hypothesis: the optimal structure of the decentralized web and the actual structure of the cortex follow the **Loki Principle**

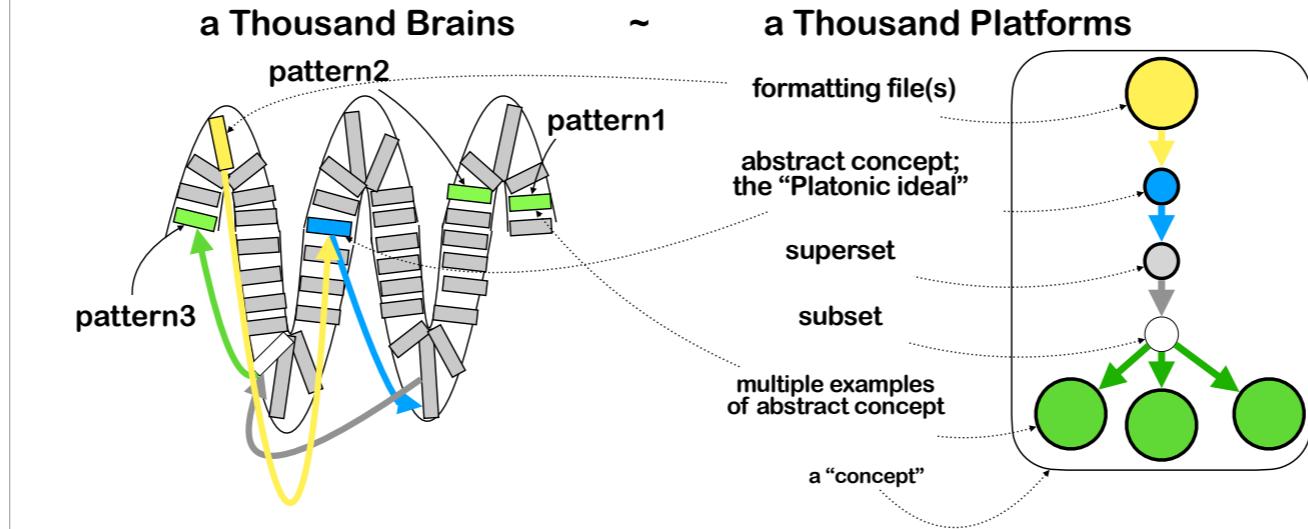
- Each has a standardized way to connect data with its format



That means that each cortical column on the left is analogous to an individual node in the graph on the right, which of course represents one data file. Just as a Loki Pathway connects a piece of data to its format in the CG, there is a similar pathway, formed by neurons, connecting one cortical column to another. And that a Loki Pathway in the cortex means the same thing that it does in the concept graph: the column at one end of any given Loki Pathway provides formatting information for the column at the other end of the Loki Pathway.

Hypothesis: the optimal structure of the decentralized web and the actual structure of the cortex **follow the Loki Principle**

- Each has a standardized way to connect data with its format
- Each implements a grapevine-like system to select from available options



And lastly, I speculate: that the method that the brain uses to pick one column over any of its rivals is a grapevine-like method. What exactly that means, I'm not sure. Is there one grapevine in, let's say, a healthy person's brain? Or are there many that exist in parallel, as is the case with the decentralized web, where there is a unique grapevine for each user? Well, perhaps the title of the book, the Thousand Brains hypothesis, could be restated as the Thousand Grapevines Hypothesis. Many of them would be expected to overlap, perhaps significantly, but they're not all exactly identical.

a Thousand Brains ~ a Thousand Platforms

Each is characterized by:

- many entities existing in parallel**
- decentralized => no one to establish consensus**

And that's it. That's the idea. The cortex and the dWeb are each decentralized systems, with many entities existing in parallel;

a Thousand Brains ~ a Thousand Platforms

Each is characterized by:

- many entities existing in parallel**
- decentralized => no one to establish consensus**
- consensus required for interoperability**

in each case, there is no single entity that is in charge of all the rest, which hinders interoperability;

a Thousand Brains ~ a Thousand Platforms

Each is characterized by:

- many entities existing in parallel
- decentralized => no one to establish consensus
- consensus required for interoperability
- The Loki Principle: the ONLY WAY to establish consensus
 - a concept graph-like system to organize data
 - a grapevine-like system to select among multiple options

and in each case, communication is only possible bc they are each organized according to the Loki Principle, which establishes Loose Consensus via a concept graph-like system to organize data and a grapevine-like system to select among multiple competing options whenever it is necessary to do so. If this is correct I might conjecture that the cortical grapevine would have significant connections to the limbic system, more so than the cortical concept graph.

[end 28]

29

Implications for the brain?

[start 29]

If my ideas about the Loki Principle and the brain true — and i admit they are merely conjectures; I offer no evidence in this video that it's true — then there are likely pretty profound implications regarding how the brain functions.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

Here are some of the questions that I think we could make progress, if my hypothesis is true.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

On the topic of knowledge encoding: I would make the observation that the act of forming a concept, fleshing it out with properties and examples, relating it to preexisting concepts either as subsets or supersets or example of or via connections to individual properties that feed into the JSON Schema — this process is tantamount to defining something. When we talk about defining a word, these steps are what we are referring to. In other words, definitions reside within the topology of the concept graph. Or at least, it seems that way to me.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

On the question of the creative process — I wonder: how do analogies work in the brain? If data is organized as a CG, then perhaps an analogy entails looking at large-scale patterns within some large chunk of the CG, encompassing a large number of individual concepts, and looking to see whether that same pattern exists in some other area of the concept graph, and whether a comparison of the patterns could be enlightening. As I mentioned earlier, looking for complex patterns in large graphs can be very computationally expensive; perhaps this explains why artists, mathematicians, writers, scientists, philosophers, etc spend so much time wandering around with their heads in the clouds. They're doing computationally expensive pattern searches and they sometimes need to be able to run without interruption.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

Regarding the functions of sleep: as I built earlier version of the CG app, it occurred to me that sometimes the relationships of sets and subsets could require optimization, cleaning up. There is evidence that functional networks in the brain exhibit small world network characteristics, and I have seen it hypothesized that one function of SWS could be to restore small world properties which become degraded during wakefulness. From the CG model, it becomes easier to understand what exactly that might mean.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

REM sleep probably serves a different function. In the grapevine, trust is transitive. And among a large group of users, there will likely be self-reinforcing, closed loops. The process of calculating influence scores requires it to be an iterative process. Because of that, managing the grapevine is unavoidably a computationally expensive endeavor. Perhaps REM sleep is an opportunity for our brain to re-tune its inner grapevine. When you dream you are fine tuning your value system.

So to summarize: perhaps SWS is when we optimize the CG, and REM sleep is when we optimize the grapevine.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

Dunbar's number has been estimated to be about 150, and it's the upper limit of the number of people we can keep track of in our heads in a social system. Well if the brain uses a grapevine-like system, and if that system requires computationally expensive fine tuning, perhaps Dunbar's number is set bc that's the most that our inner grapevine is able to handle, due to limitations of cognitive resources, that perhaps we could calculate, supposing that we were to elucidate the structural makeup, topology, and characteristics of the cortical concept graph.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

And lastly, regarding self-deception: branching from neurology to psychiatry; I have two thoughts on how this might manifest itself in a Loki-like system. First, recall that there are two ways to store the same information: one is in a file, and the other is in the topology in the graph. And the reason for using both of these systems simultaneously is that if we were to store information ONLY in the topology, it would require too much computation to look up pieces of data, and more efficient to be able to find it stored conveniently within a single file. This raises the possibility of data mismatch.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

And my second thought has to do with the notion of having multiple, perhaps conflicting, definitions for value-laden concepts, like justice, which I mentioned earlier. If your inner grapevine switches back and forth strategically between two or more conflicting definitions, then you can use this as a tactic, not only to deceive others, but also to deceive yourself, if you keep no record of the switching.

Implications for the brain?

How is knowledge encoded in the brain?

What is the creative process?

What are the functions of sleep?

Why does the brain use so much energy - even during sleep?

Dunbar's number?

How (and why) are we so good at self-deception?

And that's all I'm going to say about that now. It's obviously a complex topic, and like I said: this is pure speculation.

[end 29]

30

SUMMARY / CONCLUSIONS

[start 30]

In summary: The Principle of Loki states

SUMMARY / CONCLUSIONS

data



that for any given piece of data

SUMMARY / CONCLUSIONS

data



format



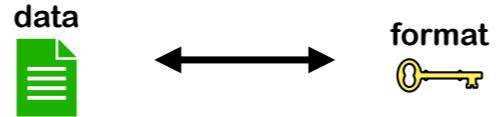
there is a format.

SUMMARY / CONCLUSIONS



And there must be a simple, straightforward, explicit way to connect the one to the other.

SUMMARY / CONCLUSIONS

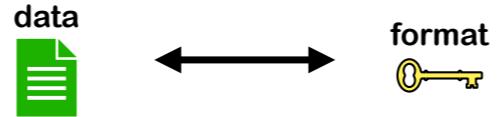


The Principle of Loki is NECESSARY for the decentralized web to reach its full potential

The cortex (maybe) ALREADY USES the Principle of Loki

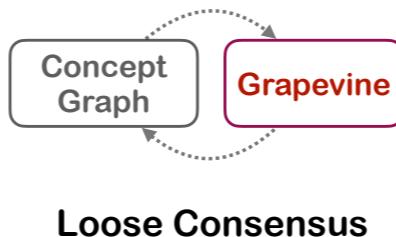
I have argued that the Principle of Loki is NECESSARY for the dWeb to reach its full potential; and that the cerebral cortex may ALREADY use the principle of Loki.

SUMMARY / CONCLUSIONS



The Principle of Loki is NECESSARY for the decentralized web to reach its full potential

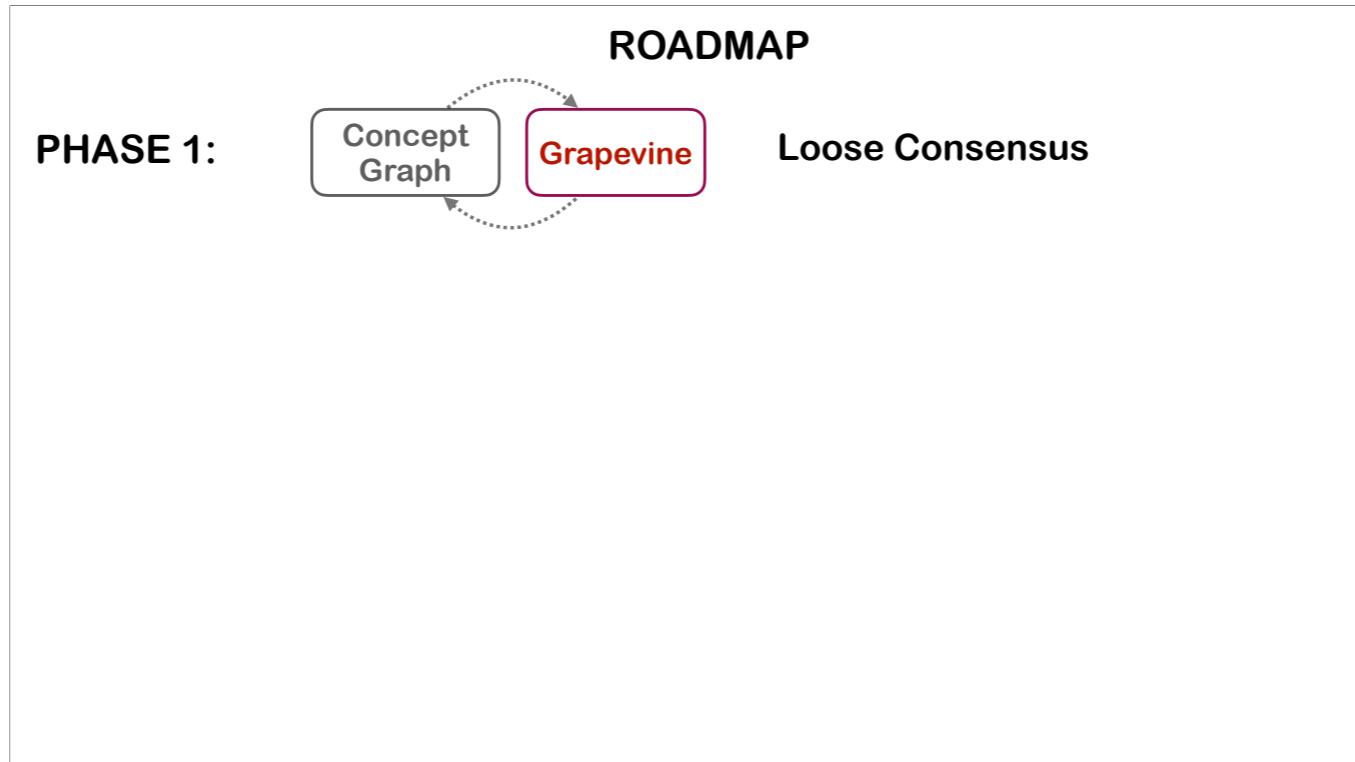
The cortex (maybe) ALREADY USES the Principle of Loki



I have described an implementation of this Principle in the form of the Concept Graph and the Grapevine, and that these two apps form the basis of Loose Consensus, which is the tool we must use if we are to tear down the Walled Gardens that prevent distinct entities such as users from effective communication and interaction in the absence of a centralized authority.

ROADMAP

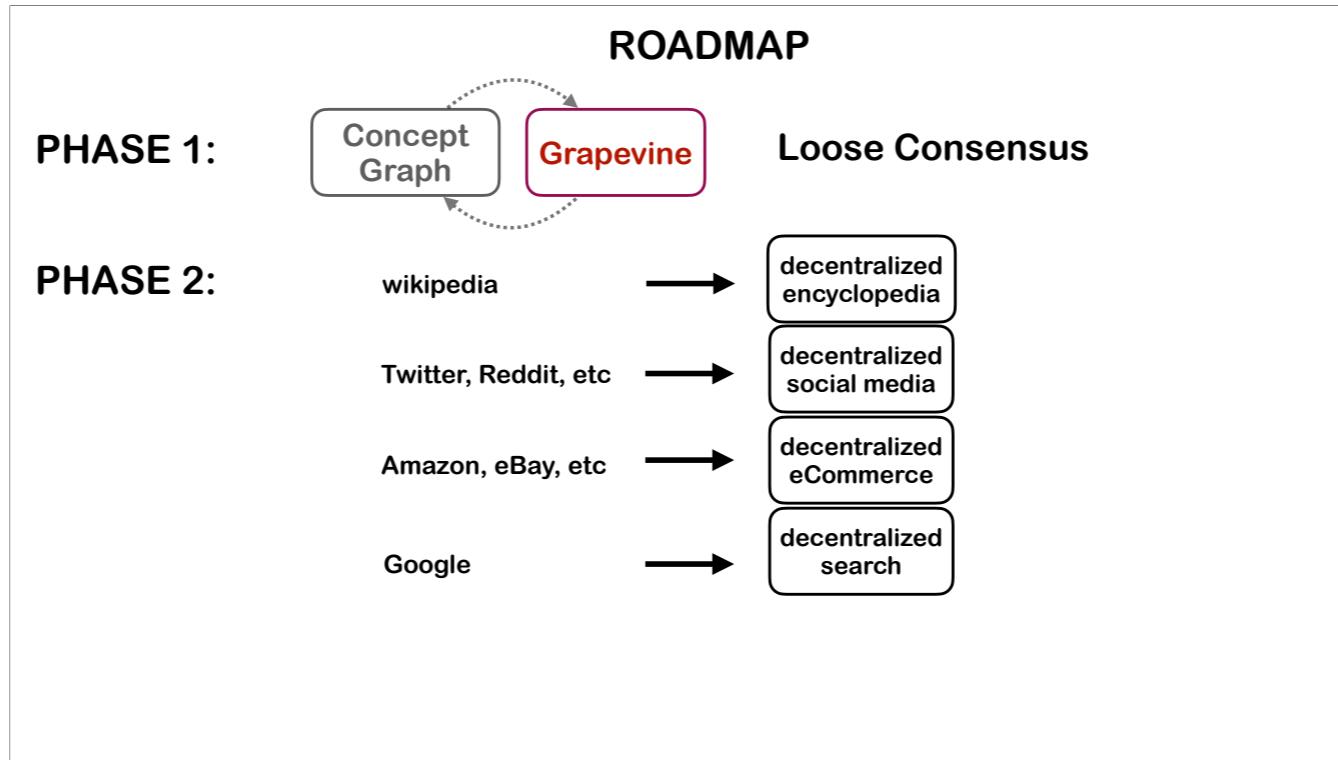
My roadmap is as follows.



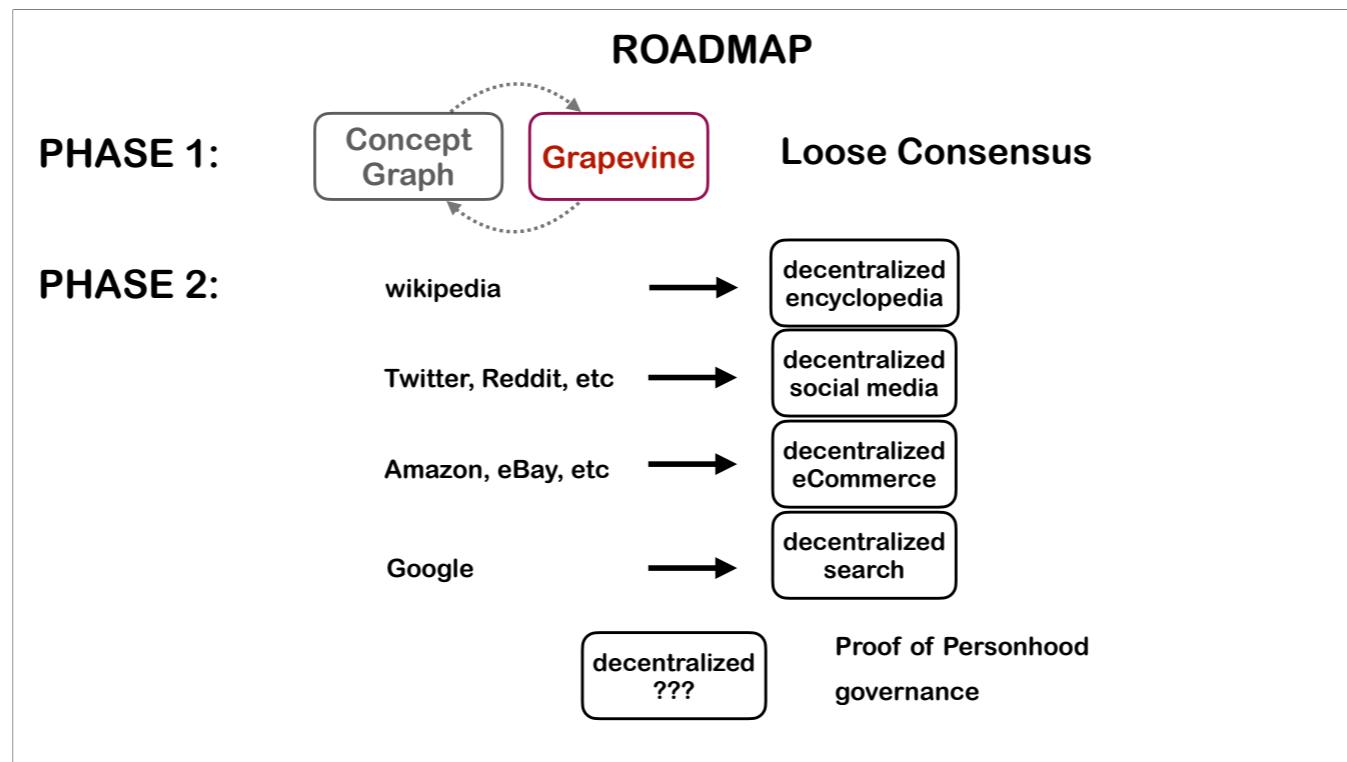
Phase 1: ReConstruction of the Concept Graph and Grapevine apps, and using them to demonstrate Loose Consensus.



Phase 2: to rebuild existing proprietary and centralized platforms



as decentralized open source apps, using the Principle of Loki and Loose Consensus as the foundation.



And lest I forget: to build new types of apps and solve problems that simply cannot be solved without the decentralized web.

Imagine the dWeb WITHOUT the Principle of Loki

To underscore the necessity of this roadmap, I'd like you to imagine the decentralized web without the Concept Graph and the Grapevine; without any implementation whatsoever of the Principle of Loki.

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Every piece of data has a format.

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Where is formatting information located?

Where is formatting information located?

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Where is formatting information located?

- proprietary platforms
- open source platforms
- standards bodies, e.g. W3C Consortium

Consider 3 possibilities.

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Where is formatting information located?

- proprietary platforms
- open source platforms
- standards bodies, e.g. W3C Consortium

The first possibility is the status quo: formatting information is the purview of proprietary platforms, like Twitter, etc. Obviously that's not what we want.

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Where is formatting information located?

- proprietary platforms
- open source platforms
- standards bodies, e.g. W3C Consortium

The second possibility: formatting information is maintained by the developers who maintain an open source projects in repos like Github. In other words, it is stored in the best of all possible worlds, in documentation; but more realistically, in the heads of developers. This is not good enough.

Imagine the dWeb WITHOUT the Principle of Loki



data



format

Where is formatting information located?

- proprietary platforms
- open source platforms
- standards bodies, e.g. W3C Consortium

Third possibility: standards bodies, such as the W3C Consortium. This is not ideal, for the simple reason that UNIVERSAL STANDARDS IS NEVER THE ANSWER when decentralization is the goal.

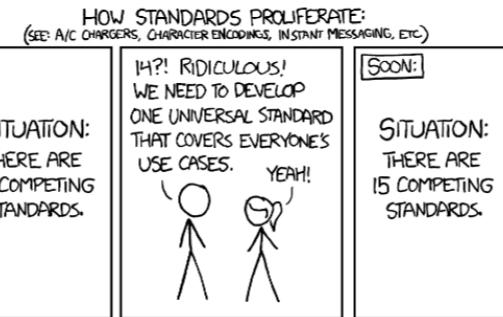
Imagine the dWeb WITHOUT the Principle of Loki



Where is formatting information located?

- proprietary platforms
- open source platforms
- standards bodies, e.g. W3C Consortium

XKCD



This point is driven home by the well known XKCD comic on How Standards Proliferate.

The situation is that there are 14 competing standards. This is noted to be ridiculous! We need to develop a SINGLE UNIVERSAL STANDARD to cover everyone's use cases! And the result, of course: 15 competing standards. Which means: the Walled Gardens remain.

PHASE 1:



W3C Consortium

> 1300 Specifications

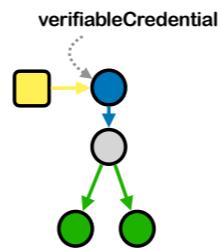
Verifiable Credentials

<https://www.w3.org/TR/?version=latest>

<https://www.w3.org/TR/2022/REC-vc-data-model-20220303/>

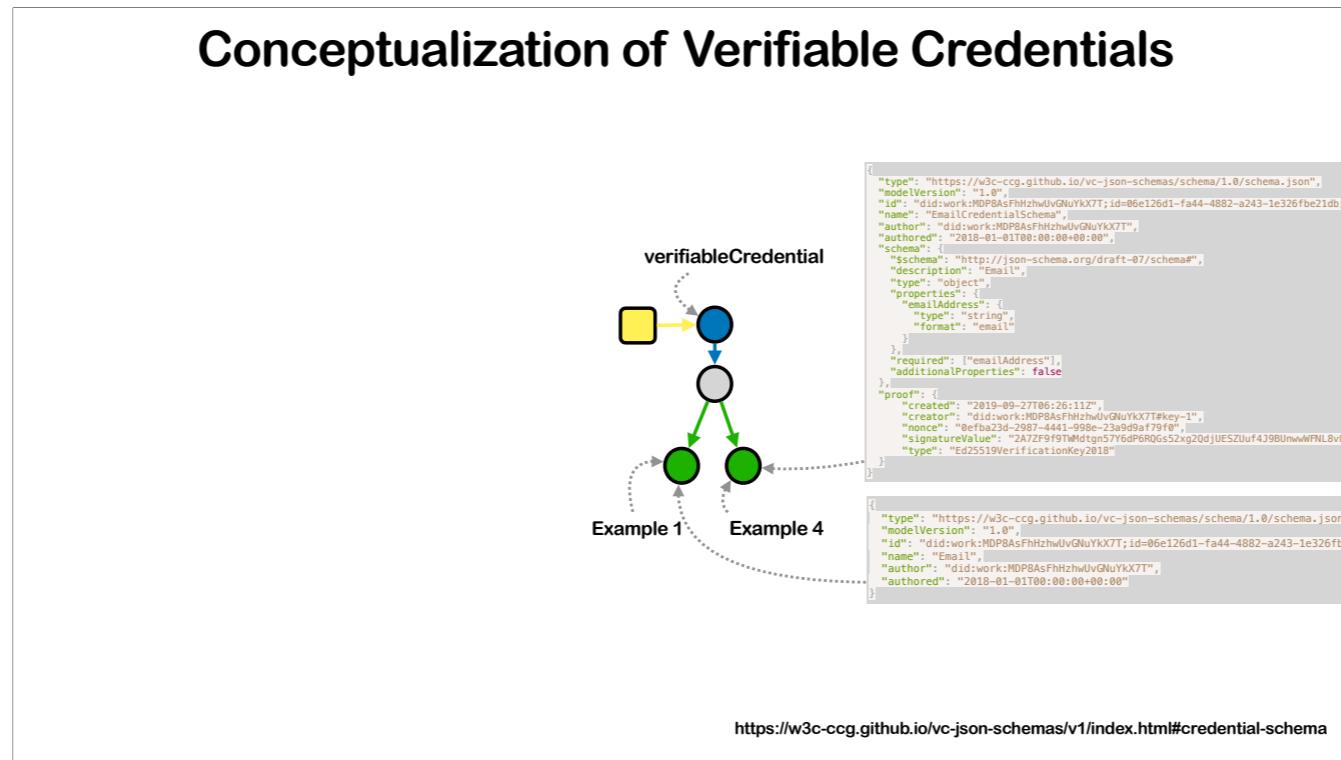
Indeed, it would seem to me that the most fertile ground for the demonstration of Loose Consensus, and for the demonstration of the utility of the CG and the Grapevine, would be for members of the W3C Consortium to use these two apps in the construction of specifications. The last I checked, there were over 1300 specifications being managed by the w3c Consortium. A particularly relevant specification for our purposes would be the Verifiable Credentials data model.

Conceptualization of Verifiable Credentials



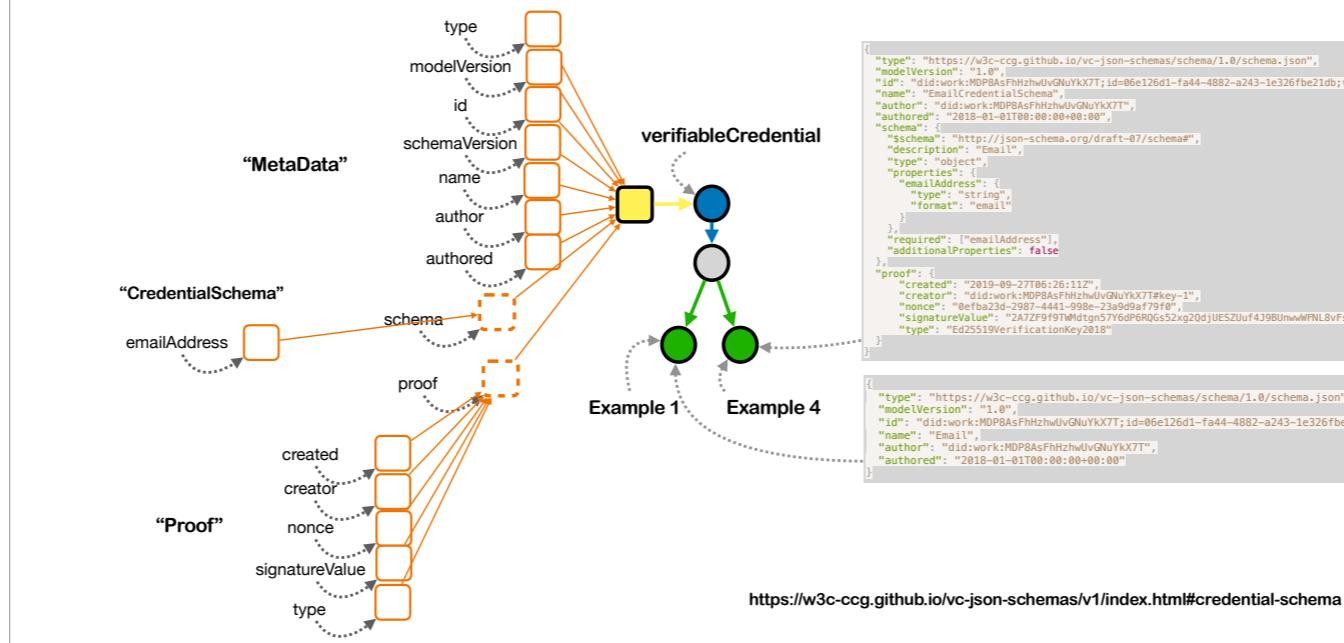
Conceptualization of the Verifiable Credentials data model would look something like this. First, there is the concept of the Verifiable Credential,

Conceptualization of Verifiable Credentials



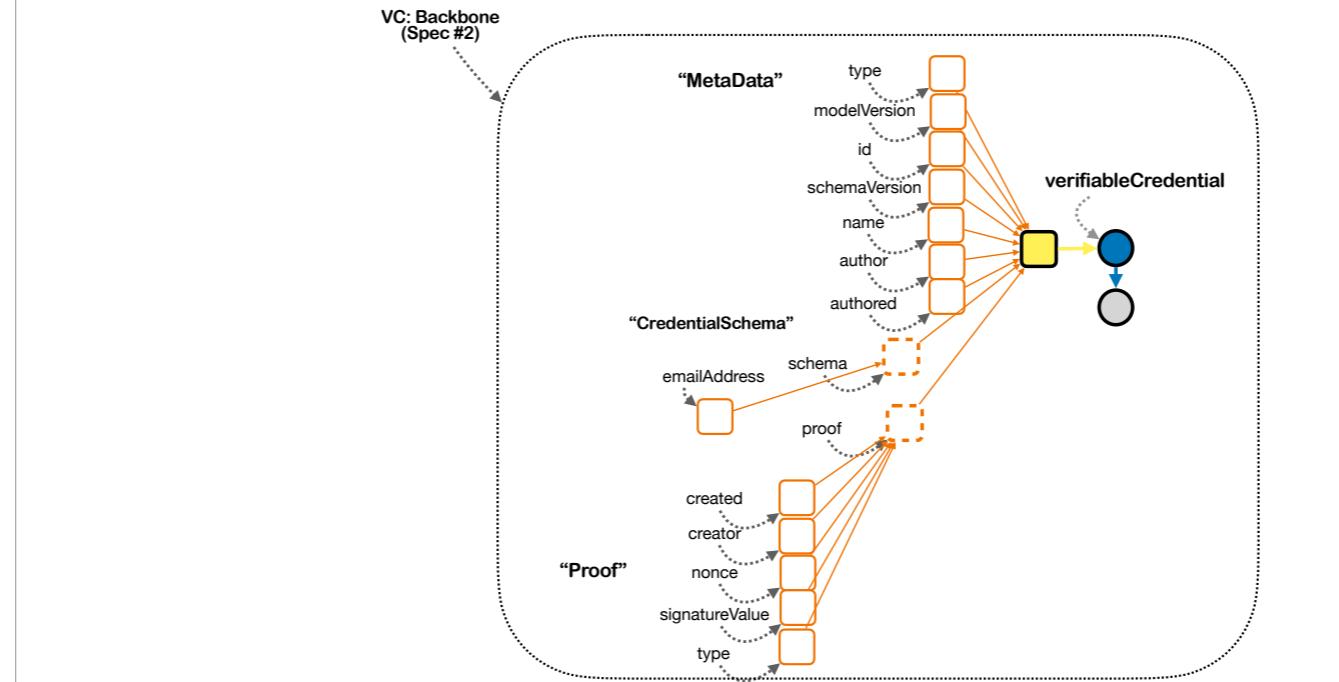
with examples that can be pulled from the W3C website.

Conceptualization of Verifiable Credentials



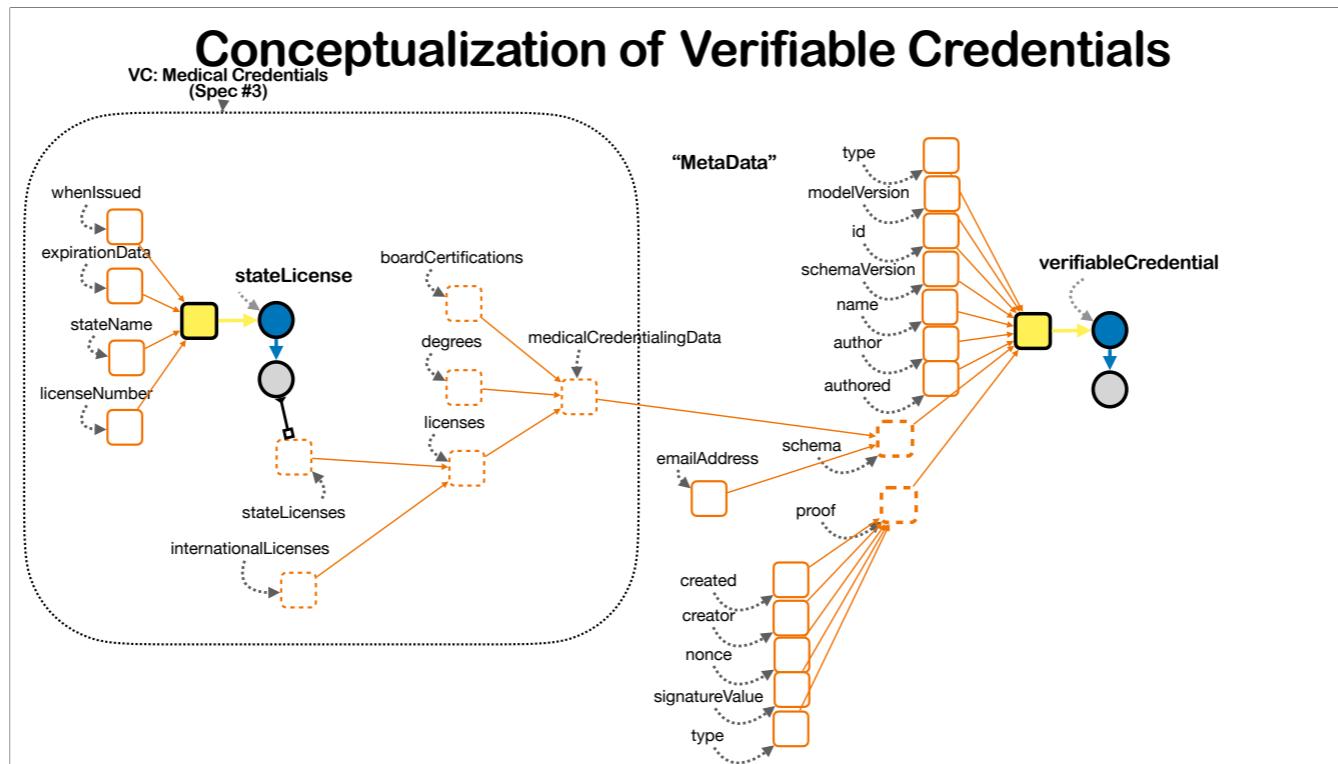
Next we have the Property Tree for the concept.

Conceptualization of Verifiable Credentials



Moving it over to the side here ...

Loose Consensus can be used to establish the backbone for the Property Tree for Verifiable Credentials. The Property Tree that I have depicted here would correspond to VC data model v1.1. At least, my understanding of it.



Then addenda to the backbone could be constructed for specific applications. For example, the VC for medical credentials might start out looking something like this. Whether this has been fleshed out, I don't know; this is simply my first draft. But how many different types of credentials are there? Well, I don't know. But if VC is to succeed, then it by necessity will be a VERY large number. And there's no way we are going to create Universal Standards for all of them. Or even, I dare say, for ANY of them, as we learned from the XKCD comic. Verifiable Credentials would be the perfect testing ground for Loose Consensus, via Phase 1.

PLAN

PHASE 1: LOOSE CONSENSUS for W3C STANDARDS

After Phase 1 is complete, demonstrating the feasibility and utility of the CG, Grapevine, and Loose Consensus,

PLAN

PHASE 1: LOOSE CONSENSUS for W3C STANDARDS

PHASE 2: HYBRID PLATFORMS

then we would move on to Phase 2.

[end 30]

31

PLAN

PHASE 1: LOOSE CONSENSUS for W3C STANDARDS

PHASE 2: HYBRID PLATFORMS

[start 31]

Phase 2 would involve what I call Hybrid Platforms.

PHASE 2: HYBRID PLATFORMS

A detailed explanation of Hybrid Platforms is beyond the scope of this presentation. But in a nutshell:

PHASE 2: HYBRID PLATFORMS

hybrid
platforms

proprietary, with easy interface to dWorld

By hybrid platforms, I mean:

PHASE 2: HYBRID PLATFORMS

legacy
platforms
proprietary

hybrid
platforms
proprietary, with easy interface to dWorld

something that has features of the legacy platforms that exist today,

PHASE 2: HYBRID PLATFORMS

legacy
platforms
proprietary

hybrid
platforms
proprietary, with easy interface to dWorld

decentralized
platforms
open source

and the completely decentralized platforms that are envisioned to exist and to make up the decentralized web of tomorrow.

PHASE 2: HYBRID PLATFORMS

legacy
platforms

proprietary

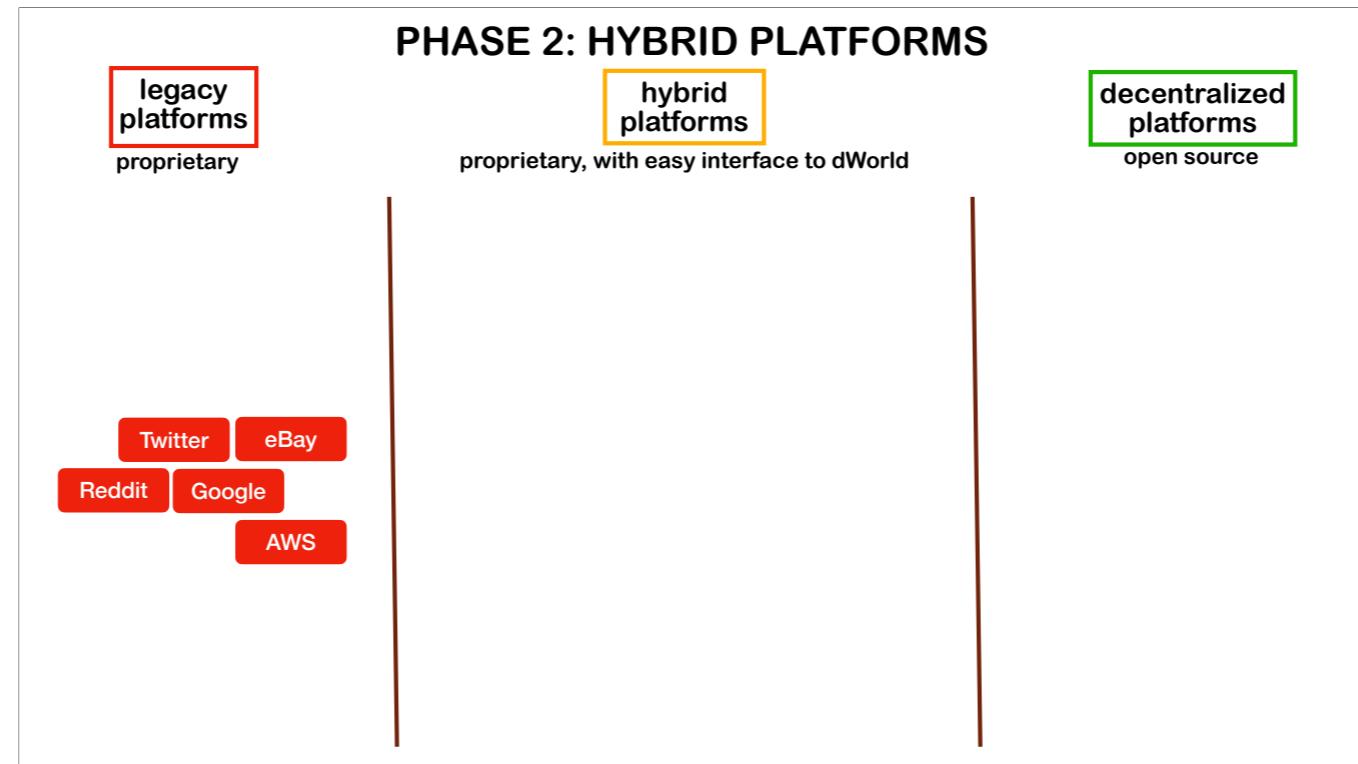
hybrid
platforms

proprietary, with easy interface to dWorld

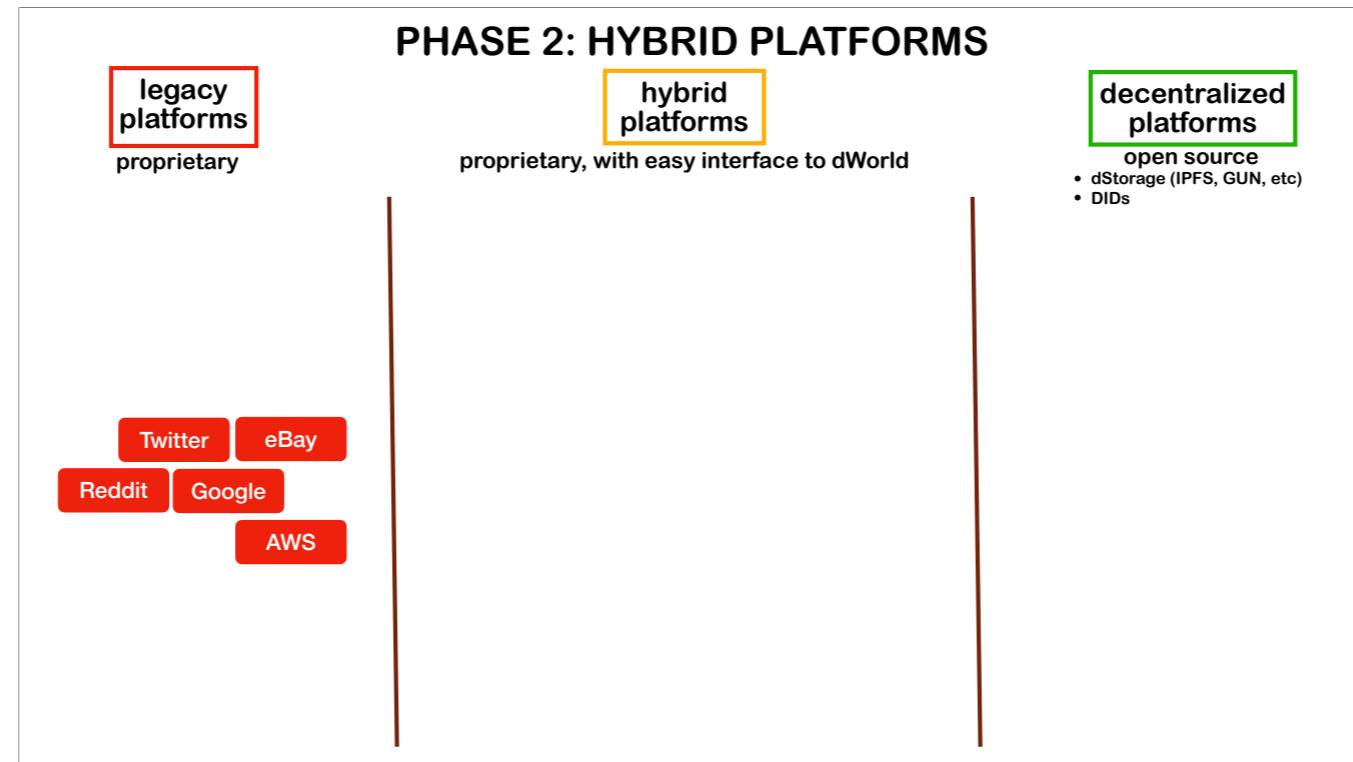
decentralized
platforms

open source

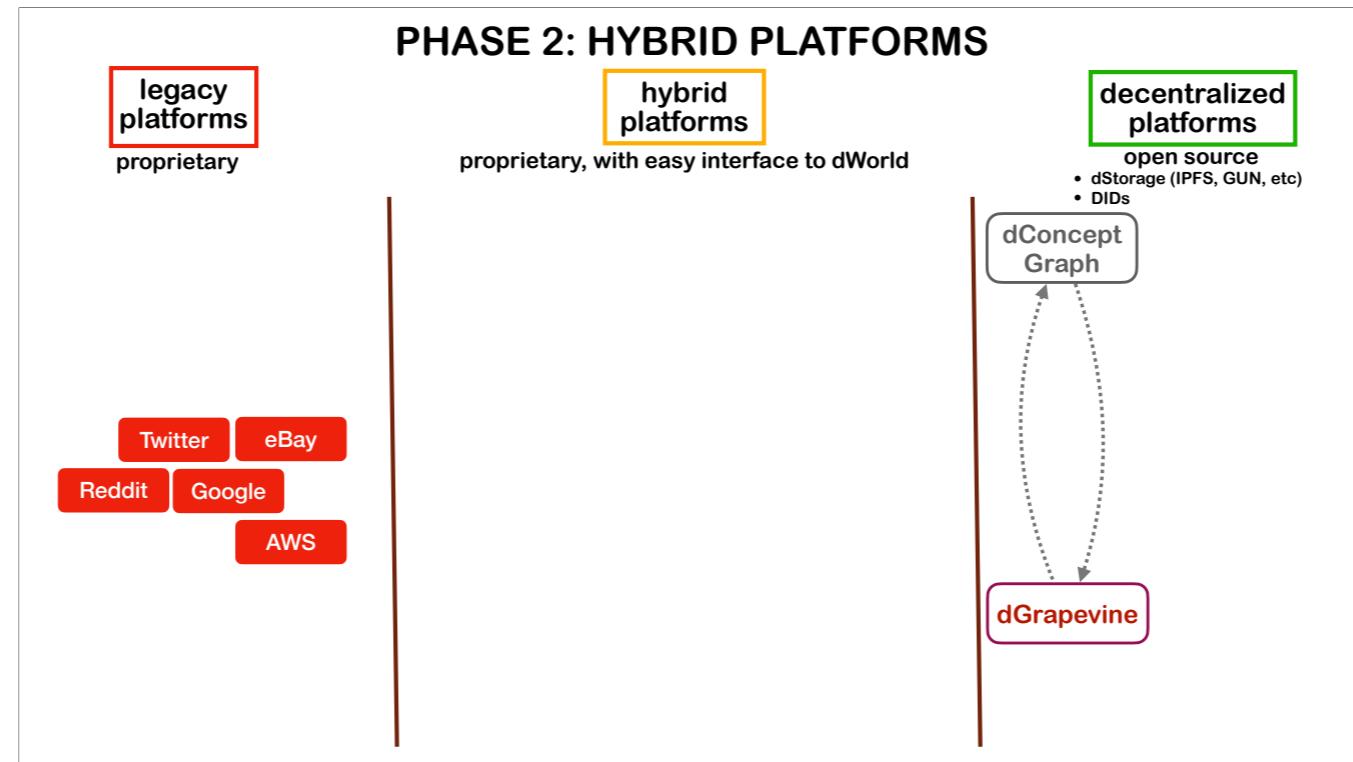
Legacy Platforms include things like



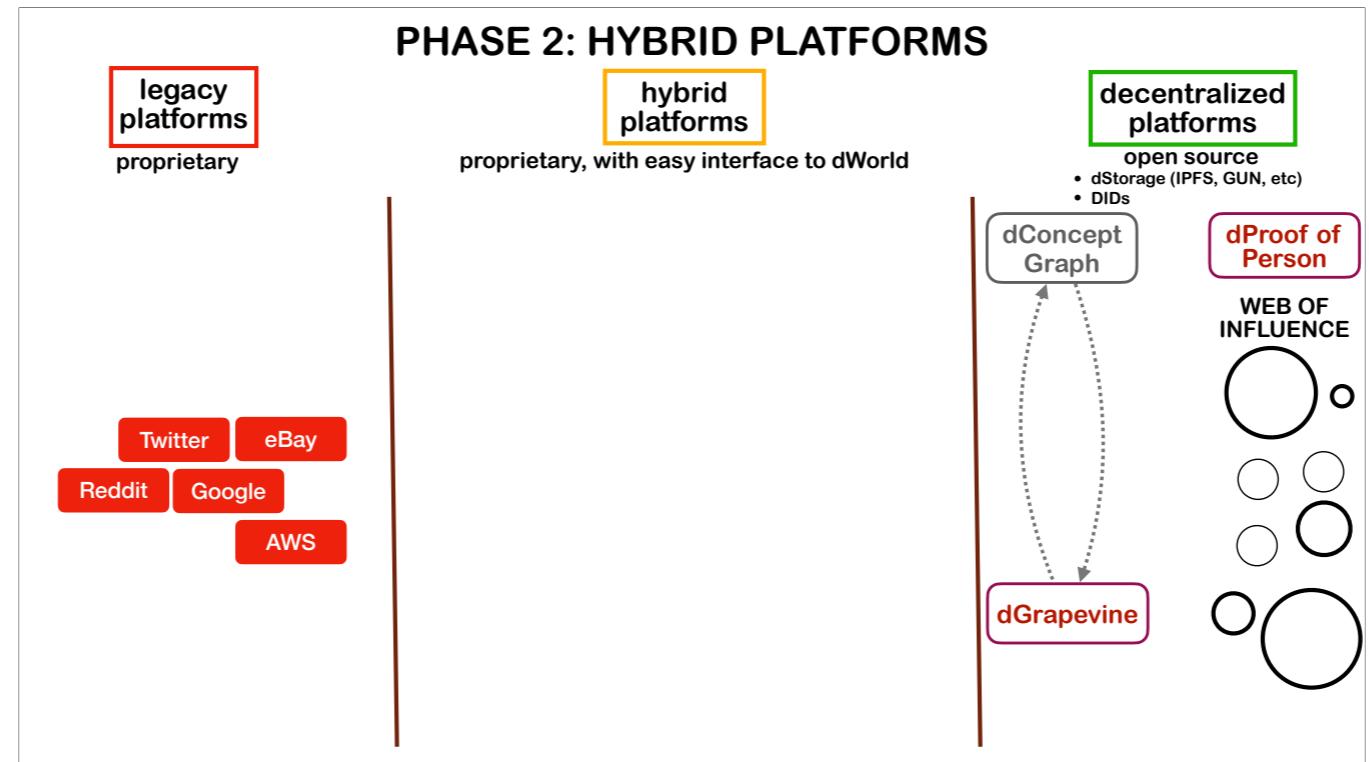
Twitter, eBay, Reddit, etc.



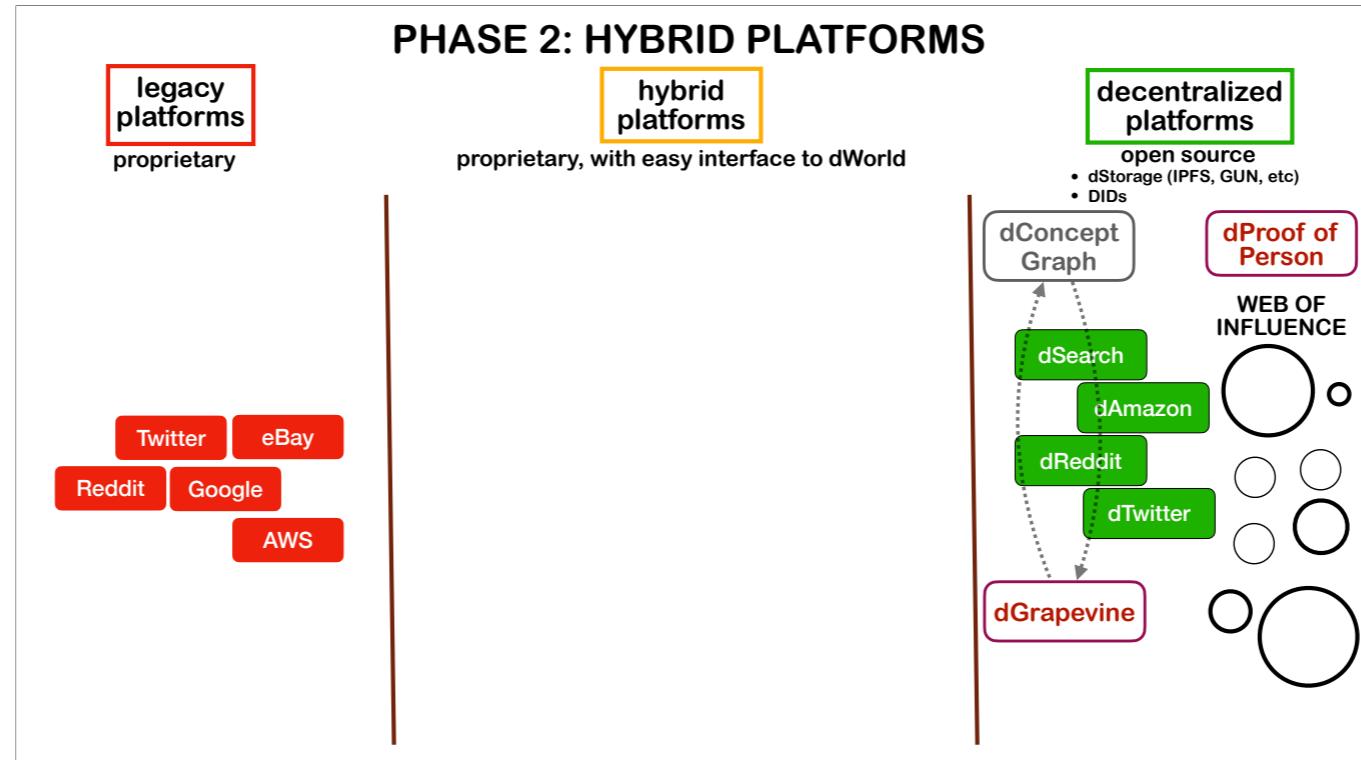
Decentralized platforms will rely upon decentralized storage, like IPFS, GUN, etc, many different types of decentralized identity tools,



They will make use of a fully open source and decentralized Concept Graph and Grapevine,

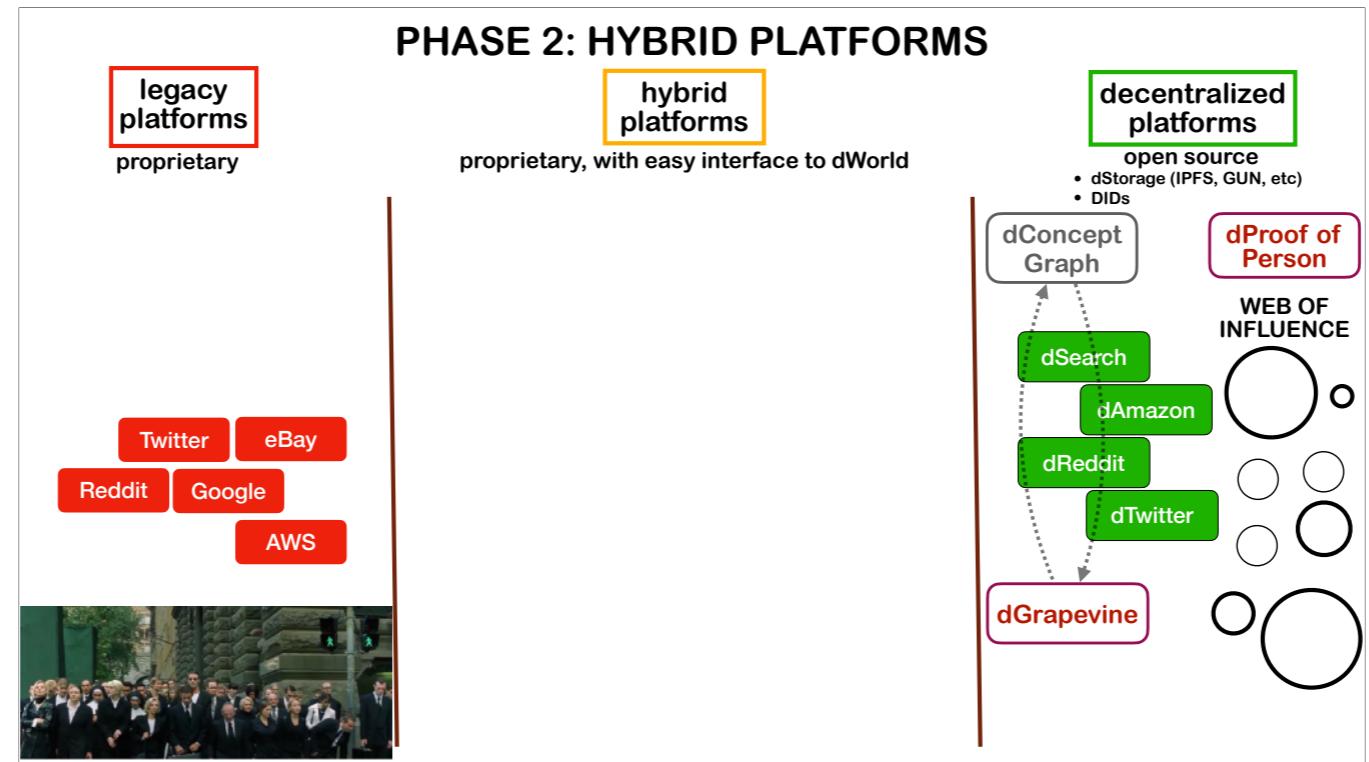


which will form the basis of a decentralized Web of Influence, decentralized Proof of Personhood,

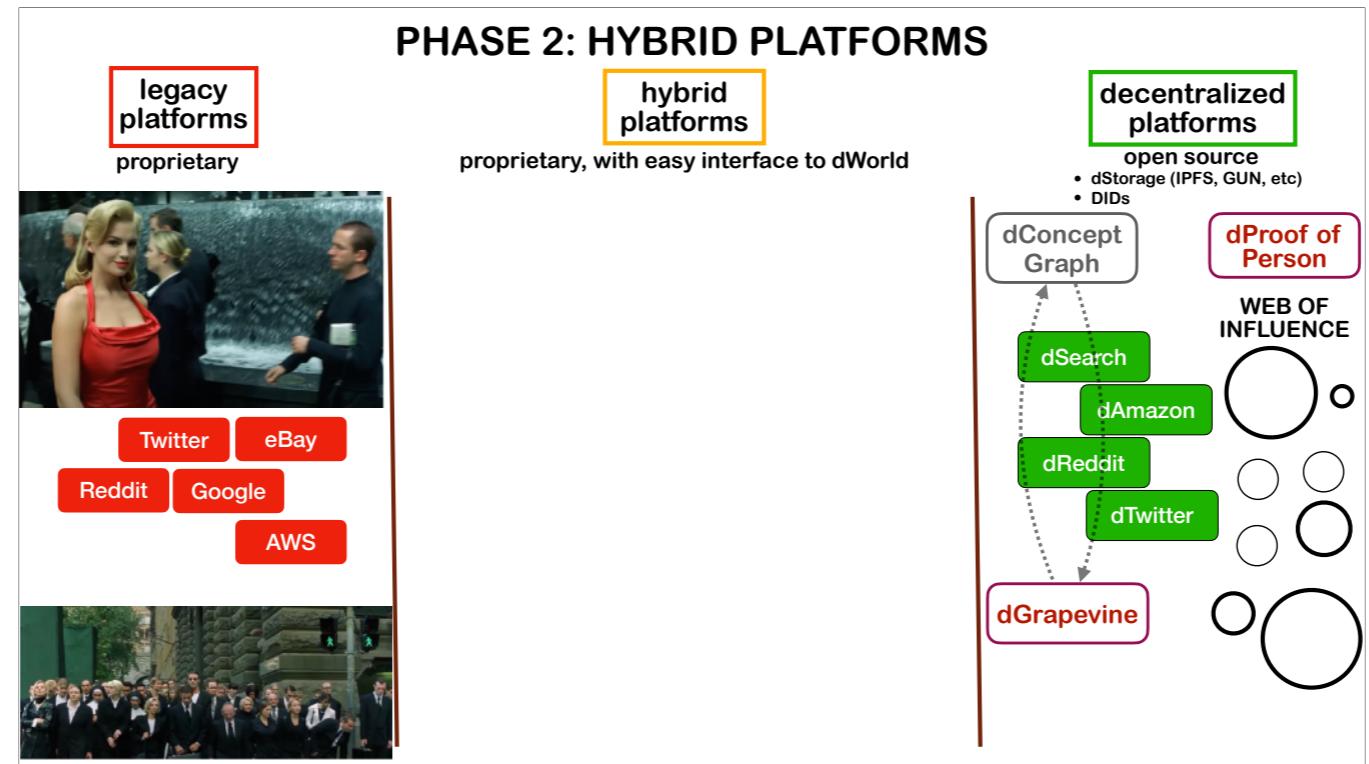


and decentralized applications for search, social media, eCommerce, etc.

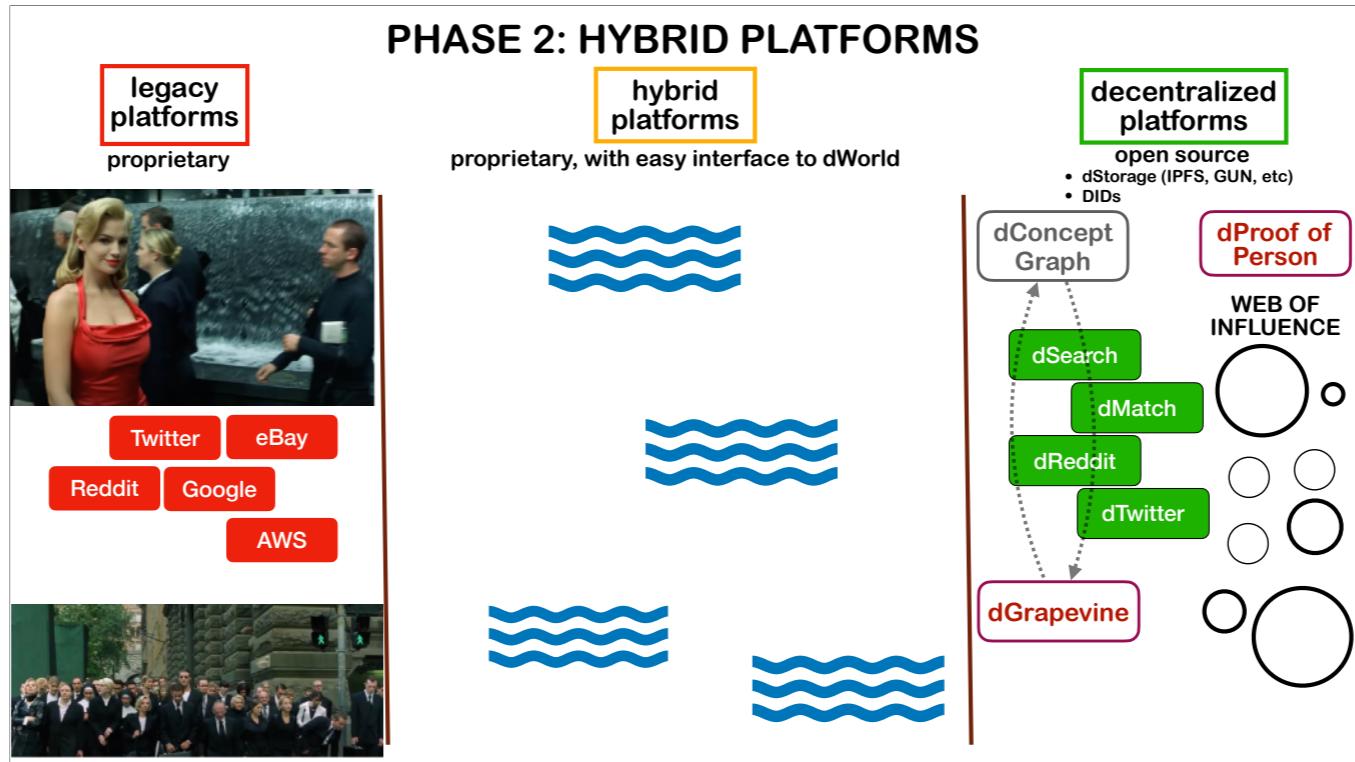
The difficulty is going to be: how do we get from here to there.



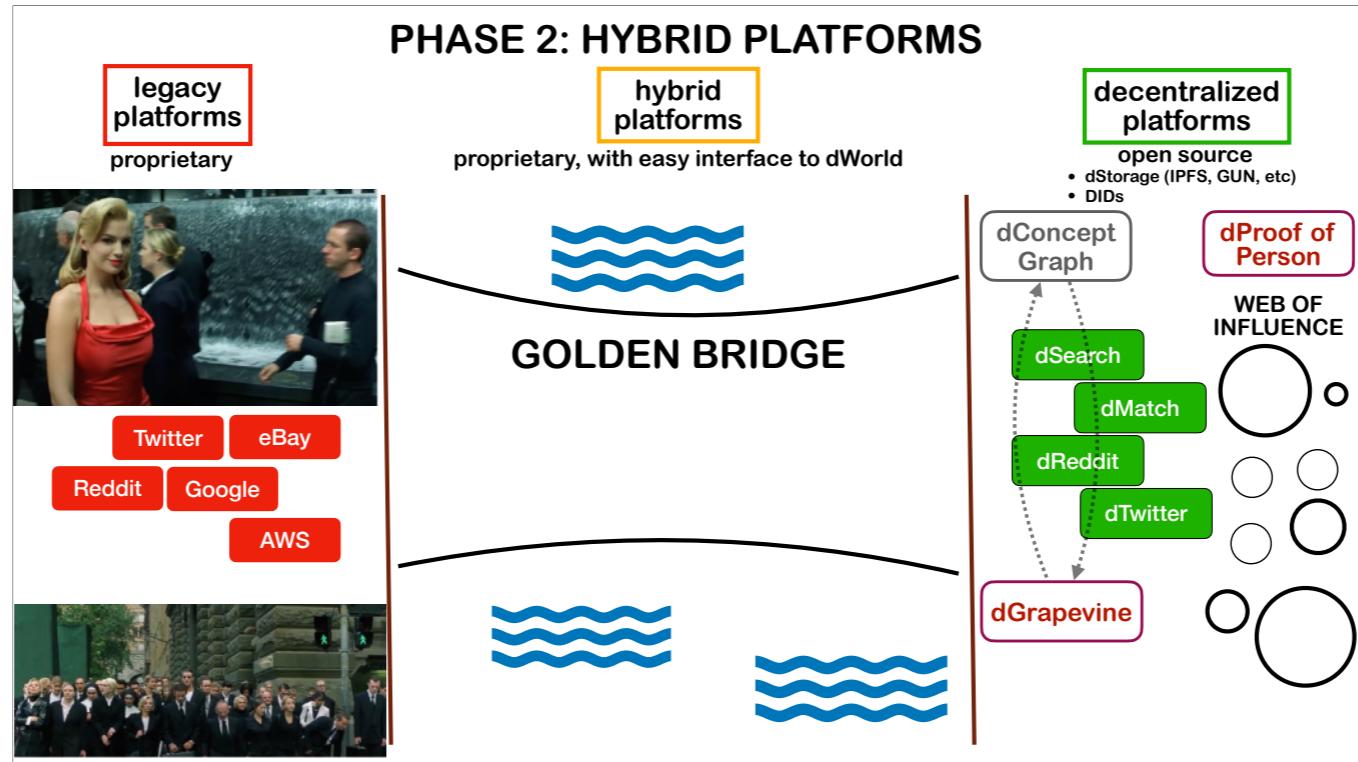
Most people are plugged into the current system, and they're not ready to wake up. They know the system has flaws,



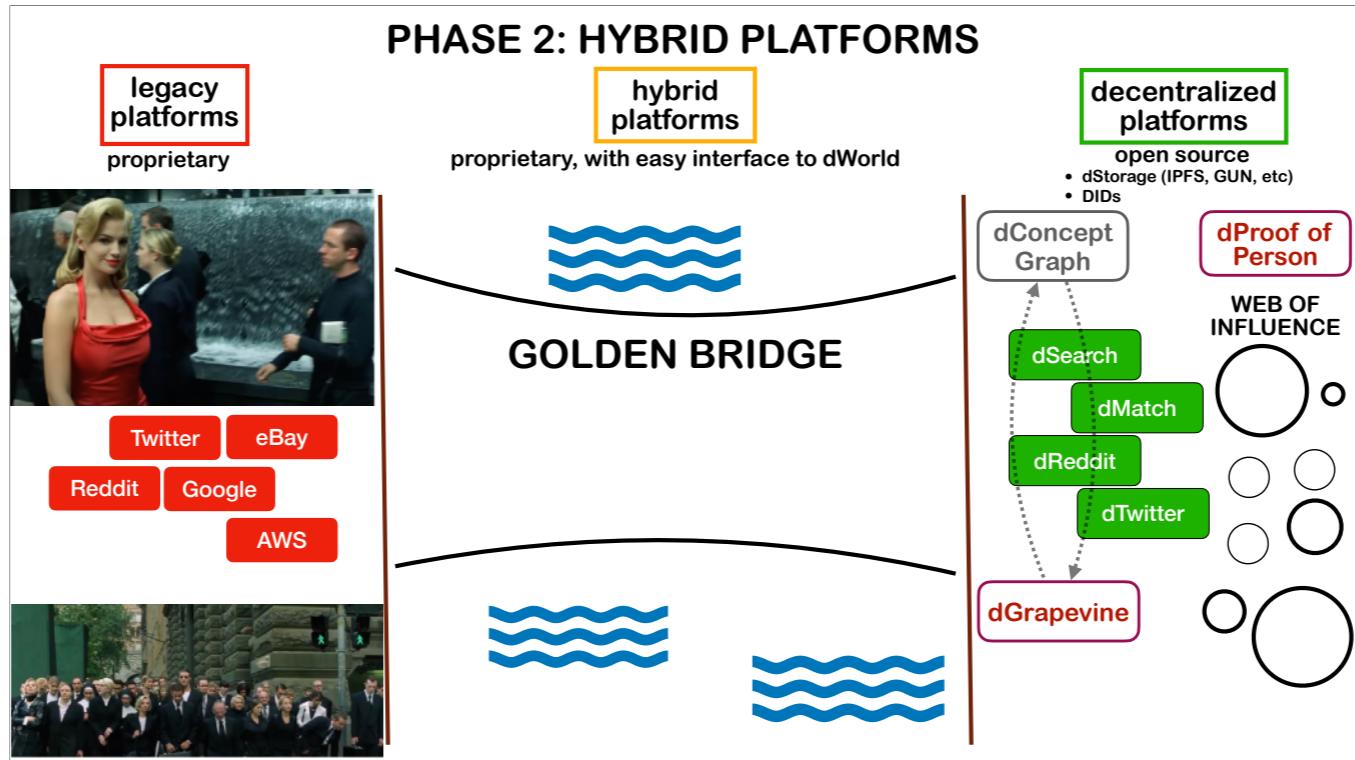
but they like it anyway and they're going to keep using it.



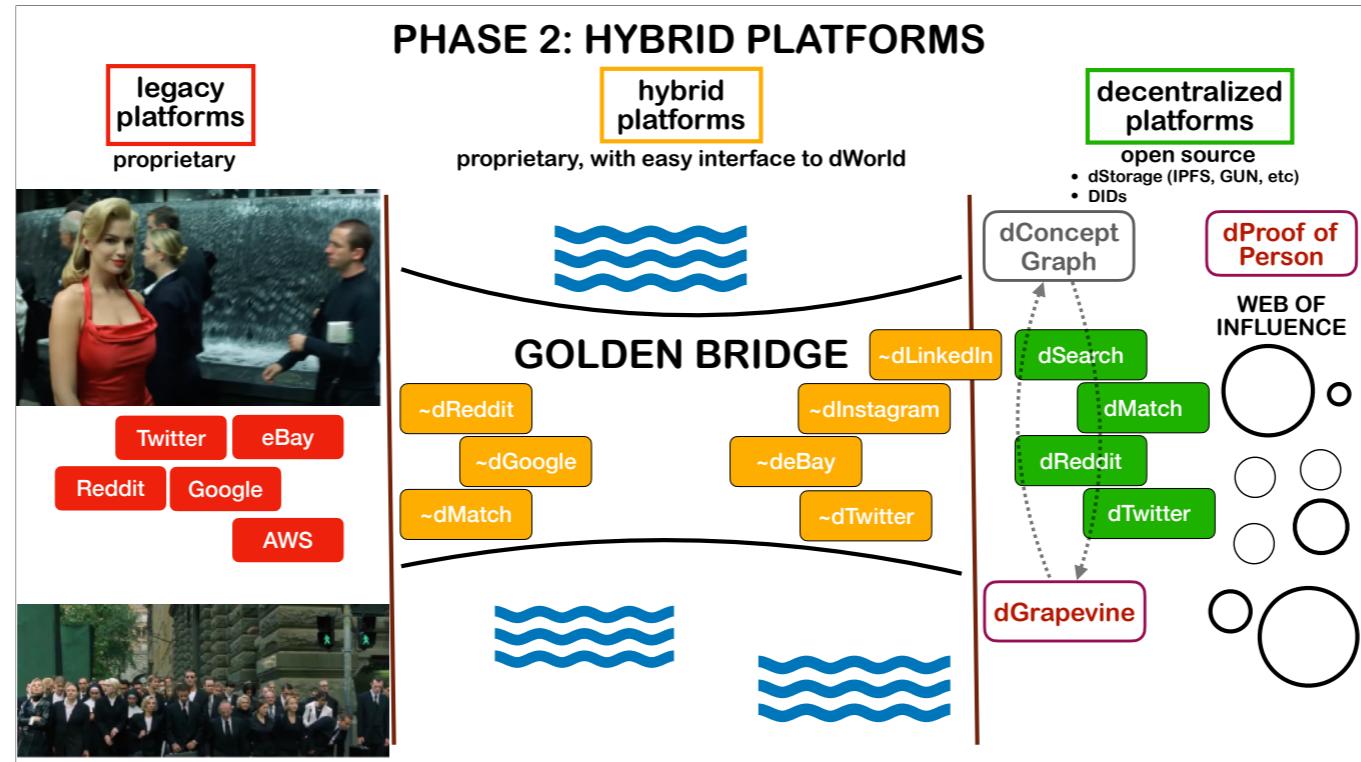
How many people are willing to cross the river so to speak to the other side? Maybe a few cypherpunk idealists. But most of the 8 billion people are not cypherpunk idealists. Most of the 8 billion people are going to need assistance. And that's what Hybrid Platforms will be for.



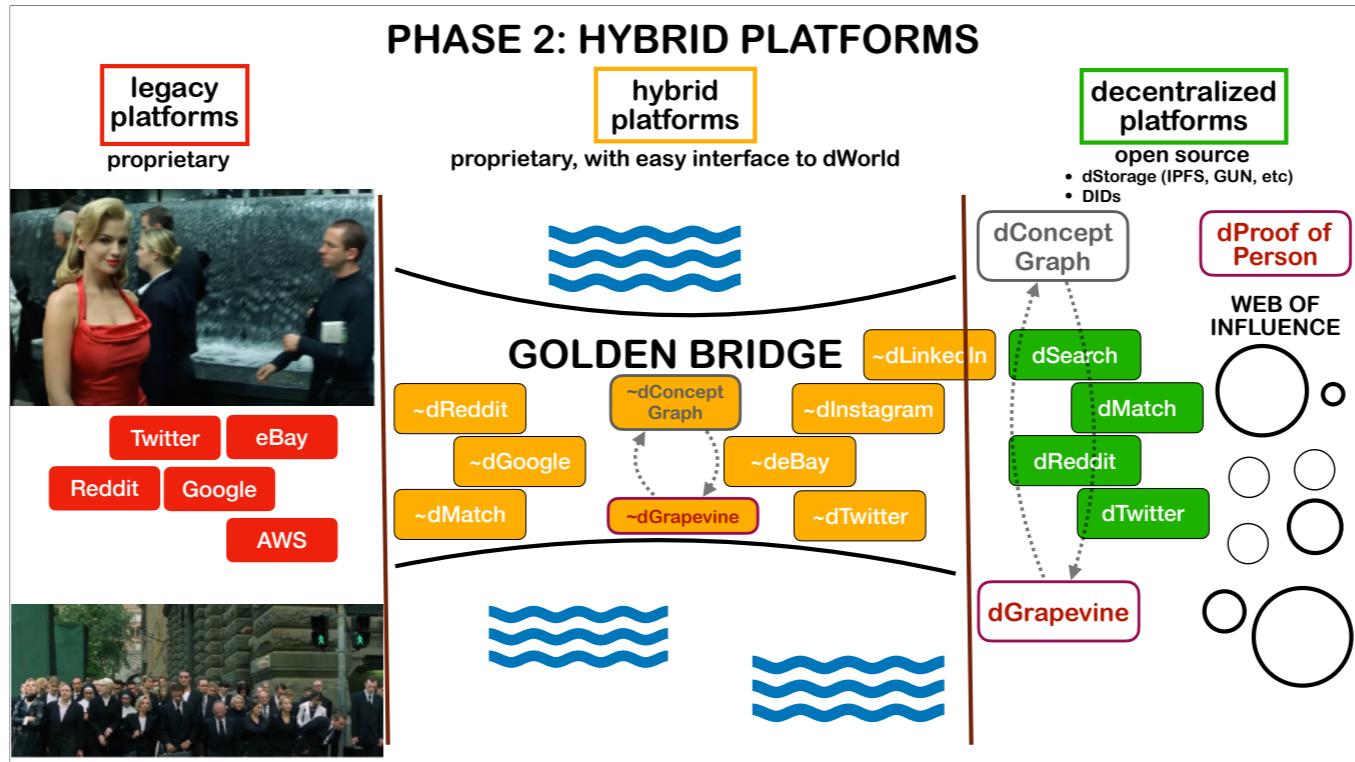
Hybrid Platforms will be the Golden Bridge from here to there.



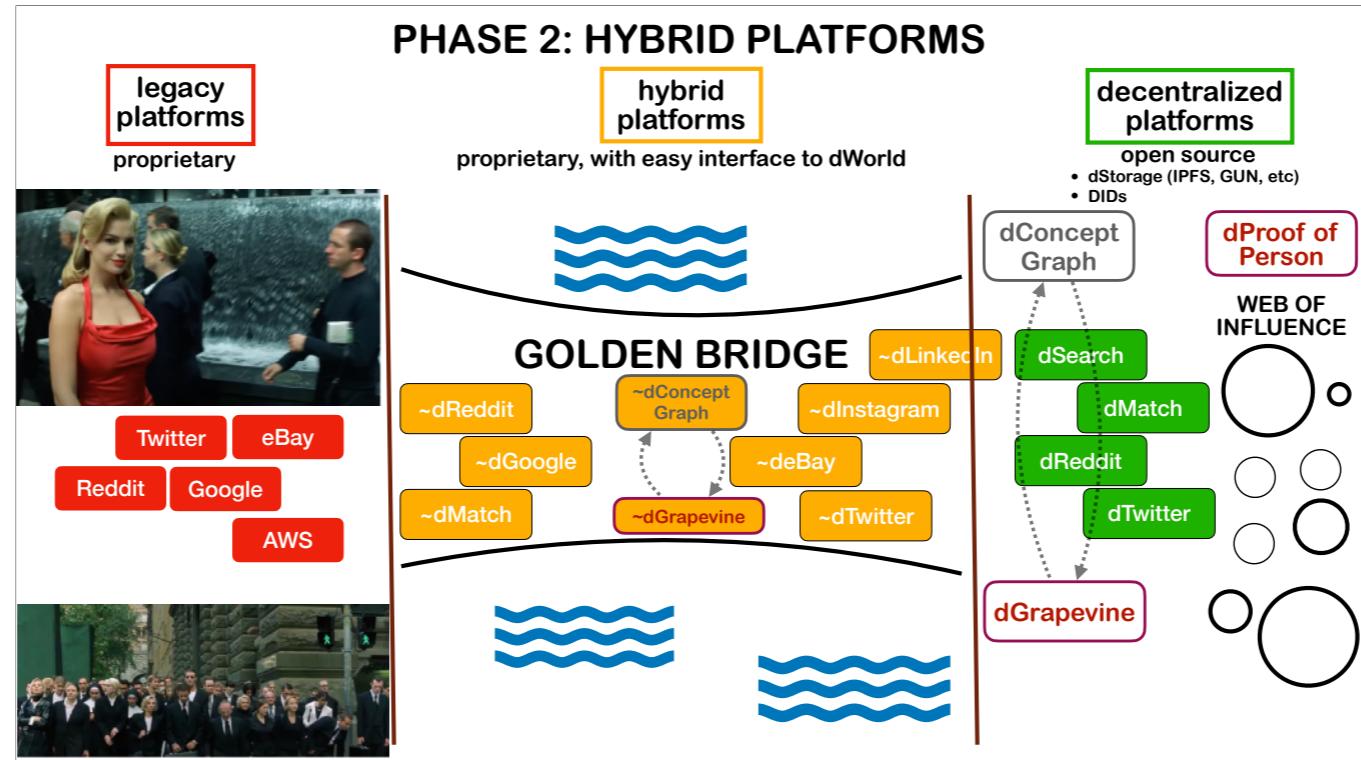
Hybrid Platforms will be proprietary, and will provide a user experience that is familiar to what everyone is already used to.



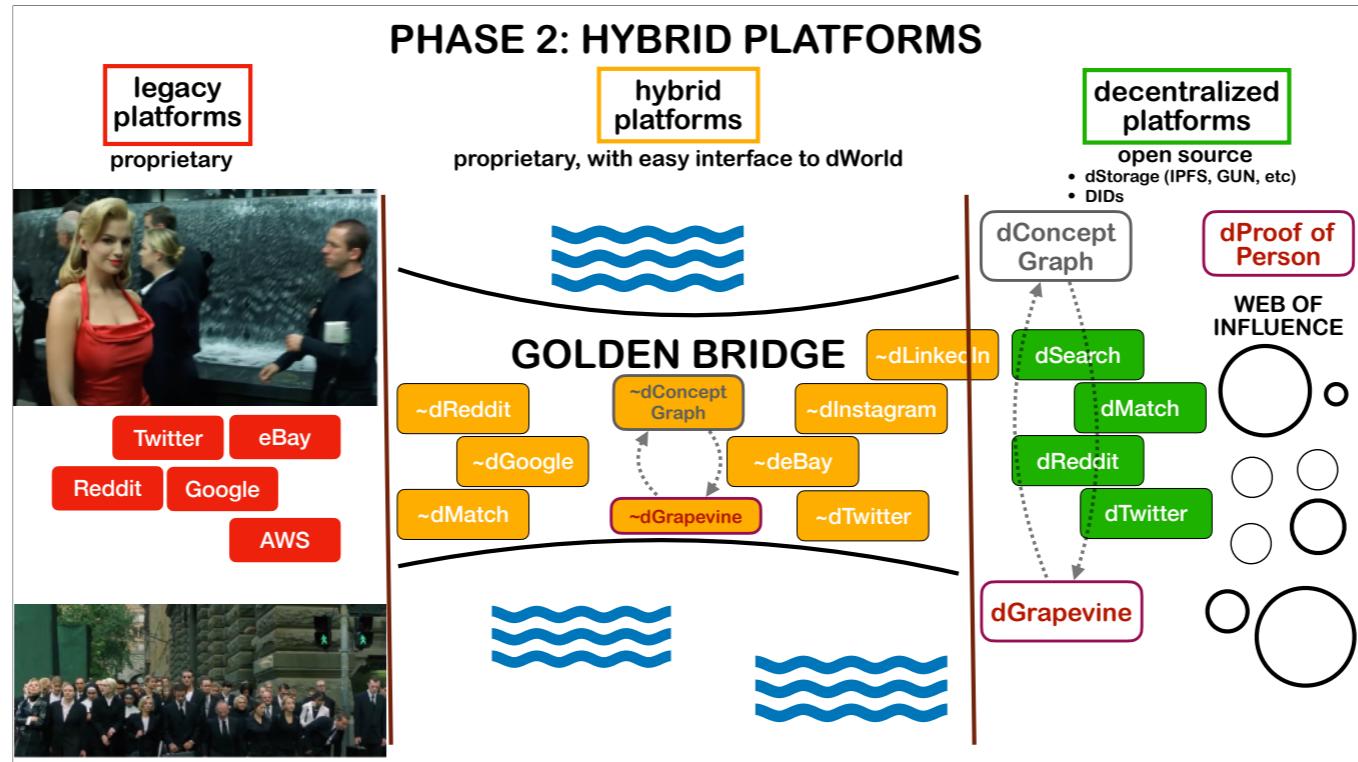
Users will go to a URL. They'll sign up. They'll have apps on their phones. All of the usual functionalities that they currently enjoy, will be there.



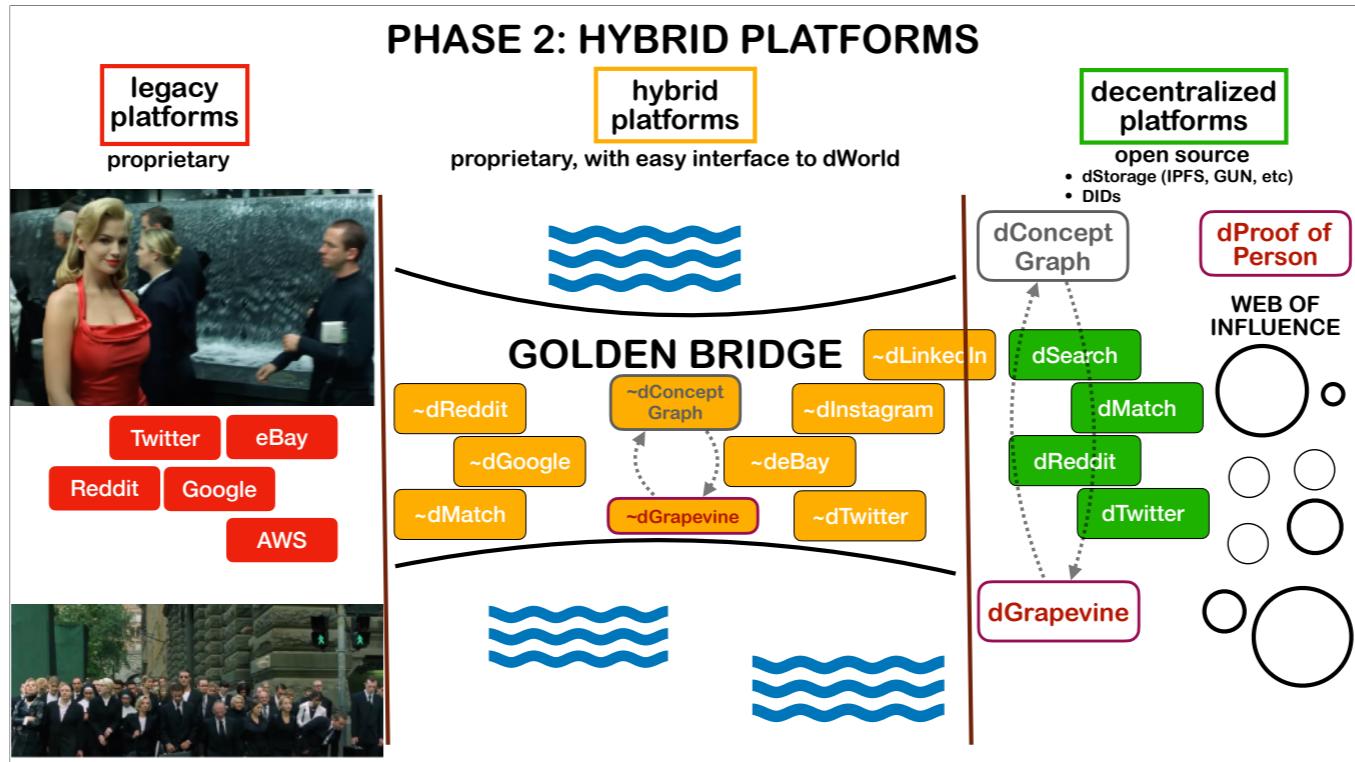
The difference is that hybrid platforms will interface with the open source world in a way that allows exit; perhaps incrementally, perhaps all at once; whenever the individual user is ready. Most of the billions of users of Hybrid Platforms may not be aware that an option like that even exists. And they might need to get specialized hardware to make it work ... too complicated. But with Hybrid Platforms, they will discover that in addition to the usual functionalities, they have this thing called the Grapevine. And that they can use it to customize ... whatever it is that they might want to customize. They won't have to if they don't want to. But some will.



And they will gradually realize that the content on these hybrid platforms is of higher quality than what they are used to. The curation is better. If they see a news item on their hybrid platform twitter, they'll have tools to help them decide whether the item comes from a trustworthy source or not. The difference, I believe, between Hybrid Platforms, and what we have now, will become like the difference between night and day. If you remember what internet search was like before google debuted in 1998, and if you remember how magically effective google was, compared to everything that came before it: that was an impressive transition. The jump from the status quo to the hybrid platform I think will be equally impressive.



Hybrid platforms may be monetized in the same way as legacy platforms. Perhaps other avenues may present themselves as well. Perhaps Hybrid Platforms will turn out to be evil, in the same way that google became evil. But the difference will be that Hybrid platforms will have a built-in exit. An exit that must always stay open, if the hybrid platforms are to maintain their value proposition. It may take time, it may take 30 years, but eventually the majority of the billions of internet users will have crossed to the other side.



The hands of most of these users will be held all along the way. As long as their hands are held, hybrid platform owners will make money. In the long run, hybrid platforms will make money, AND will save the world.

PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

So, to any builder of hybrid platforms, I would ask: what is best in life? The answer could be:

PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

- **make money**
- **save the world**

To make money. And to save the world.

And yes, those are good.

PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

- make money
- save the world



But what is BEST?

PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

- make money
- save the world

Twitter

eBay

Reddit

Google

AWS



PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

- make money
- save the world



TO CRUSH THE STATISTS. TO SEE THEM DRIVEN BEFORE YOU. AND
TO HEAR THE LAMENTATIONS OF THEIR NEVERCOINER APOLOGISTS.



THAT is best.

PHASE 2: HYBRID PLATFORMS - WHAT IS BEST?

- make money
- save the world



TO CRUSH THE STATISTS. TO SEE THEM DRIVEN BEFORE YOU. AND
TO HEAR THE LAMENTATIONS OF THEIR NEVERCOINER APOLOGISTS.



the end
[end 31]

32

INTERESTED?

[start 32]

If this project interests you, you can contact me at the email address above. I'm interested in feedback: did this make sense; it is nonsense; whatever.

I haven't yet put together a team. But it's getting close to the point where I think it may be time.

INTERESTED?

**PHASE 1: Concept Graph & Grapevine rebuilds
open-source: AGPLv3**

If you think you may be interested in helping to build Phase 1, send me a message. I could use a handful of developers.

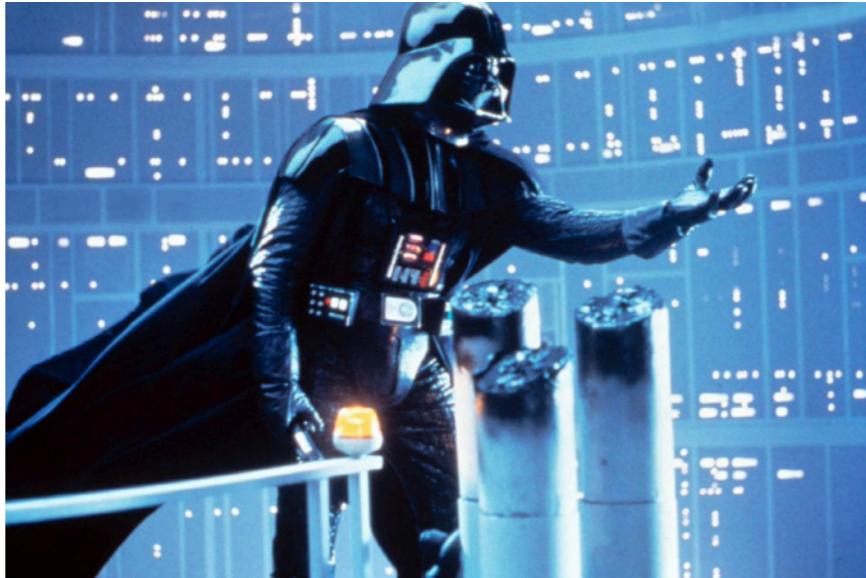
INTERESTED?

**PHASE 1: Concept Graph & Grapevine rebuilds
open-source: AGPLv3**

PHASE 2: HYBRID PLATFORMS

As for phase 2: that's going to be much more complex than phase 2. I've never run a company like this; I might be interested in a cofounder.

JOIN ME



So if you want to join me in building Hybrid platforms, with aspirations to challenge, maybe replace today's FAANG tech companies — Facebook, Amazon, Google, and the rest — beware; it may be a path to the dark side.

Principle of Loki

simple idea for a complex problem

[Move this to the end of the talk?] I also want to make the point that this idea is a very simple idea. And yet, what I want to show in this video is: that if you take this idea seriously and run with it, although it gets tedious, a lot of things follow inevitably as consequences. The stuff going on underneath the hood of the CG seems very detailed, and hard to follow, and tedious. And yet all of these tedious details flow as a natural, inevitable consequence of this one simple idea, provided you are willing to take it seriously and follow it all the way through. Also some big ideas are touched upon. Like social constructs.

Roadmap: Progress

✓ Grapevine proof of concept

✓ Concept Graph proof of concept app

Use concept graph to conceptualize:

itself:

- the grapevine
- the concept graph

test on other apps that people are working on - contact me!

- synonym, JSON-LD

- d proof of personhood

replace web 2.0:

- wikipedia

- dTwitter, dReddit, dDatingApps, etc

- dSearch (dGoogle)

- dECommerce

Inevitable because it's necessary and

Next steps

rebuild concept graph from scratch; I'll need project lead for this

use CG to "conceptualize" lots of things - to make sure it works
make it useful to 3rd parties: synonym, JSON-LD, verified credentials, etc

proof of personhood

Then use CG to replace web2.0 with decentralized:

Twitter
dating apps
reddit, for news

SEARCH - google killer

Monetization?

license Concept Graph: GNU Affero General Public License v3.0

- open source community can user Concept Graph
- but only I can implement Concept Graph on proprietary centralized platforms

PLAN

PHASE 1: LOOSE CONSENSUS for W3C STANDARDS

PHASE 2: HYBRID PLATFORMS

PHASE 1: LOOSE CONSENSUS for W3C STANDARDS

rebuild Concept Graph app (v0.1)

Use v0.1 to conceptualize:

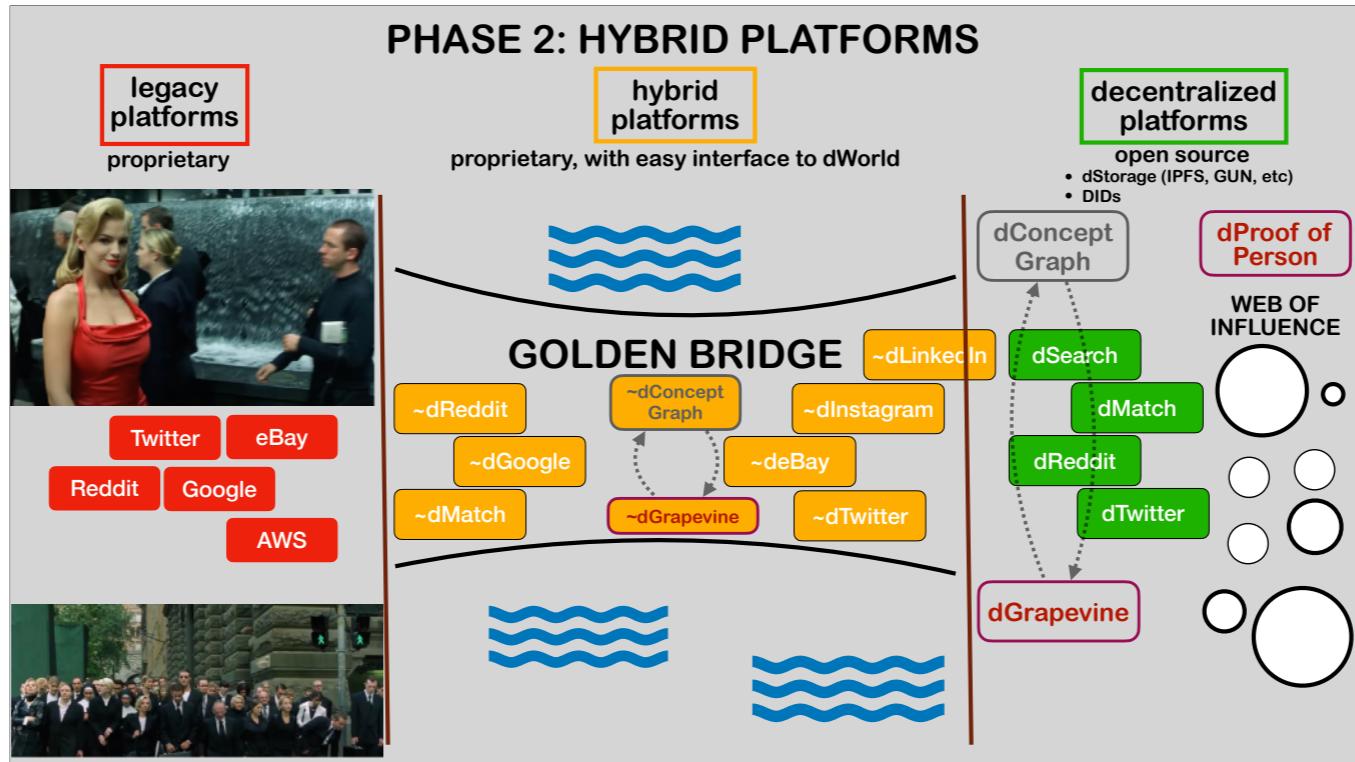
- Grapevine -> rebuild Concept Graph app (v0.2)
- Concept Graph -> rebuild Grapevine app (v0.2)

User v0.2 apps to establish LOOSE CONSENSUS on W3C Specifications:

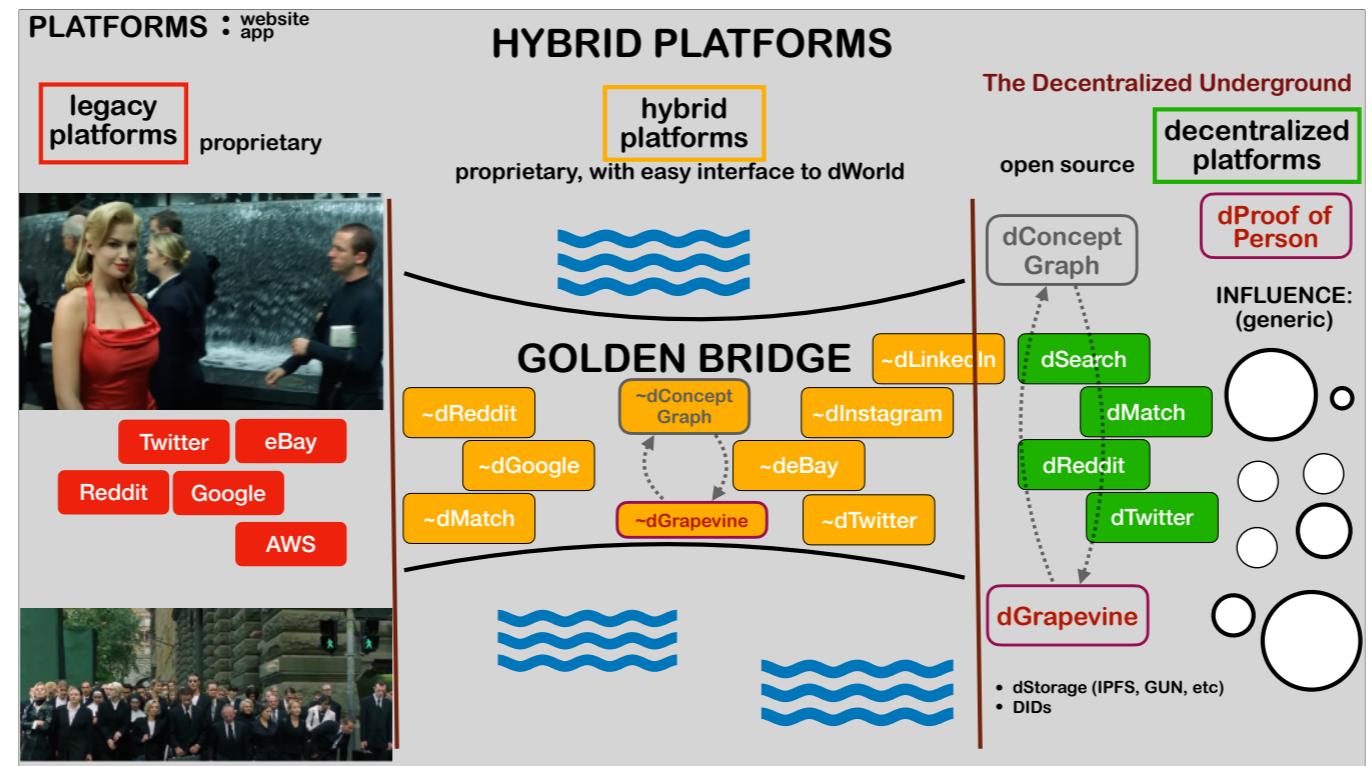
- Verifiable Credentials
- JSON- Linked Data
- 1341 Specifications listed at <https://www.w3.org/TR/>

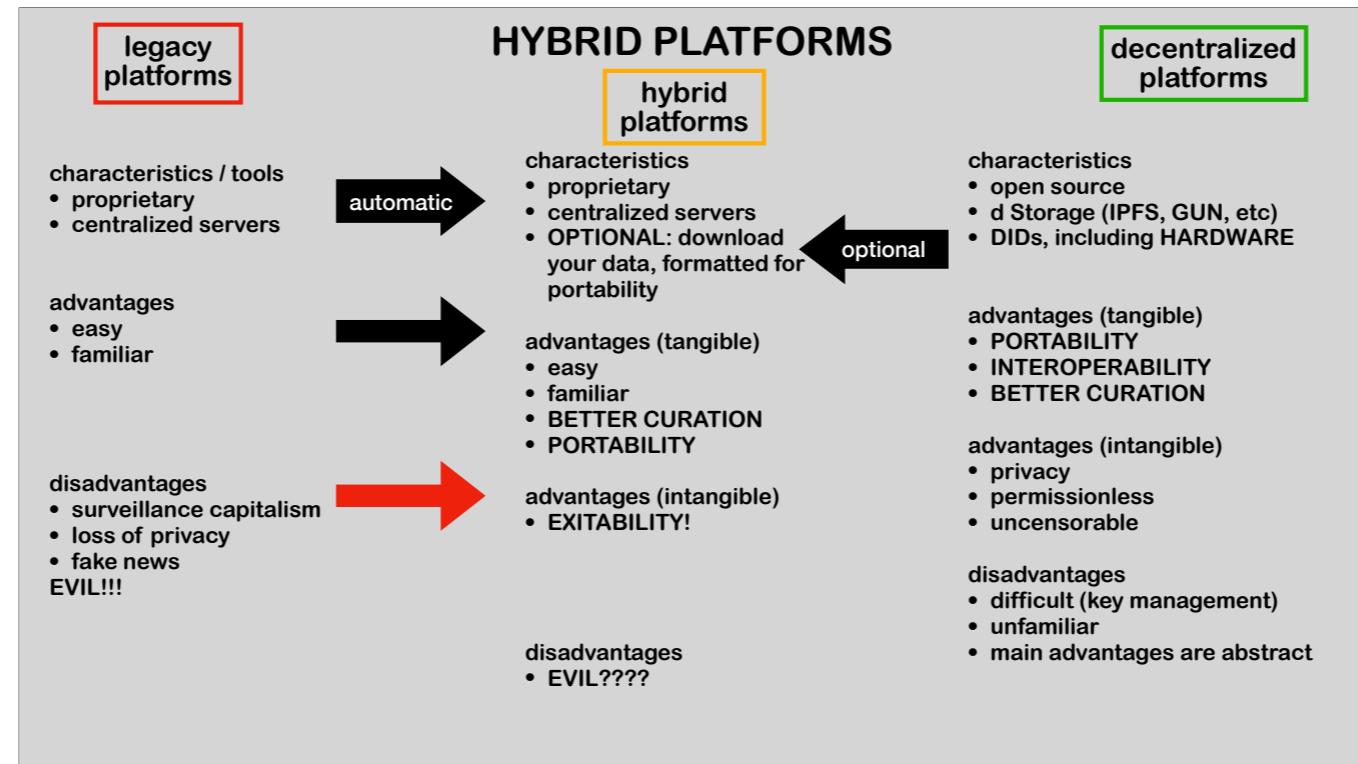
PHASE 2: HYBRID PLATFORMS

To enact all of this into a reality, we will need something that I call a Hybrid Platform. By hybrid, I mean: something that has features of the legacy platforms that exist today, and the completely decentralized platforms that are envisioned to exist and to make up the decentralized web. The dPlatforms will not require fiat entities like corporations to exist, to serve users, to evolve and to thrive. But they cannot exist without certain tools, and we cannot just jump from here to there all at once. We need a bridge from here to there. And I believe the hybrid platform is that bridge.



The dPlatforms will not require fiat entities like corporations to exist, to serve users, to evolve and to thrive. But they cannot exist without certain tools, and we cannot just jump from here to there all at once. We need a bridge from here to there. And I believe the hybrid platform is that bridge.





~dReddit

Hi Alice!

ready to EXIT?
download your data to the IPFS! get a DID!
download our app! buy hardware device!

What does my Grapevine tell me about ...

What's happening in the world
• ipsum lorem
• ipsum lorem

What's happening in Ukraine
• Did the Russians really attack a nuclear reactor?

What's happening with Covid
• Are masks effective?
• ipsum lorem

curated by:
generic
generic & Rus-Ukr war

physician-scientists

https://www.PrettyGoodApps.com/news

Why move from legacy to the decentralized web?

advantage

- better privacy
- uncensorability
- spam
- flame wars
- fake news

who cares?

- cypherpunks
- cypherpunks + people who have been deplatformed
- EVERYONE
- EVERYONE
- EVERYONE

* enough to move to a new platform

Open Question

Is JSON the way to go? (I think it is)

Is JSON Schema the way to go? (I think it is: don't know what else)

is this the best way to build JSON Schema: break them into properties?

alternate: JSON Schema Form Generator — but these are optimized for form generation, not for concept generation

HYBRID PLATFORMS

legacy
platform

Reddit

hybrid
platform

~dReddit

decentralized
platform

dReddit

- centralized

“DON’T BE EVIL”

- surveillance capitalism
- censorship
- insidious impact on attention, focus, user cognition

HYBRID PLATFORMS

legacy
platform

Reddit

hybrid
platform

~dReddit

decentralized
platform

dReddit

- hybrid legacy / decentralized
- proprietary software
- centralized servers
- UI identical to legacy platform
- BUT users have options
- add decentralized ID; digitally sign messages with devices
- export all your relevant data to smartphone / app
- **PORTABILITY and INTEROPERABILITY** with the decentralized underground ****with competitors**** are the **VALUE PROPOSITION**

TRANSITION



HYBRID PLATFORMS

legacy
platform

Reddit

hybrid
platform

~dReddit

decentralized
platform

dReddit

- decentralized

“DON’T BE EVIL”
surveillance capitalism
monopolization of attention

Decentralized Underground

dUnderground

- essential for the value proposition of hybrid platforms
- purely open source, cypherpunk in every way
- not dependent on any corporate entity

Roadmap: Pretty Good Apps

vet my strategy with other people online, at conferences
VETTING OF IDEAS: 6-12 months

find cofounder, incorporate, build team; 2M raise, angel investors (half me + half cofounder)

INITIAL HIRES 1: 6-12 months from finding cofounder

- Concept Graph project lead: 300k / year
- back-end dev
- front-end dev
- database manager (IPFS, sqlite, neo4j)
- most \$200k/year salary + options
- burn: \$1M / year

rebuild the Concept Graph 0.10 - electron app and server-side

- basics of JSON Schemas are done, including arrays and objects
- but some difficult questions, especially regarding conditionals
- user must be able to craft / edit ANY ARBITRARY JSON Schema
- test app with outside projects: JSON-LD, Verifiable Credentials, synonym, etc - if they say "this is easy to use and is useful" then ready to proceed to next step

need seamless flow:

UI, user who has no idea what JSON is

property graph

JSON Schema (alternative tools??)

MILESTONE 1: Concept Graph 0.1 release, 12-18 months after first hire

I have built this and started over from scratch at least 10 times. Ready to do it again. But this time I will need a team.

Roadmap: Pretty Good Apps

create ~dGrapevine 0.1 = made without the CG; 0.2 = made using CG (conceptualized)

- angel raise: additional \$2M raise (me + cofounder)
- hire: Grapevine project lead 300k/year
- 1-2 additional devs
- designer with expertise in graphical data display
- legal
- burn: \$2M/year
- MILESTONE 2: Grapevine 0.2 release, 6-12 months after M1

open source dConcept Graph 0.2 and dGrapevine 0.2

nurture a backbone of users for the Decentralized Underground who will use apps:

- dConcept Graph
- dGrapevine
- d???? — dReddit? dTwitter? dPoP?
- must DEMONSTRATE INTEROPERABILITY with multiple DIDs, multiple dStorage solutions, multiple dApps, multiple hardware devices
- at least 10,000 active users

keys on computer OR option to use keys from hardware device (which hardware device to start??)

MILESTONE 3: self-sufficient Decentralized Underground: 18-24 months after M2
angel raise: additional \$2M raise (me + cofounder) with goal Milestone 4

Once Decentralized Underground has some users, THEN start to launch Hybrid Platforms:

- ~dReddit? ~dTwitter? ~dSomethingElse?
- interface with apps from the Decentralized Underground
- OPTION: lease to third parties
 - angel raise: additional \$2M raise (me + cofounder)
 - MILESTONE 4: 100k users for first ~dPlatform (website): 18 months after launch
 - 10M to 100M valuation, 5-7 years after finding cofounder (100-1k per user)
 - VC raise: 10-20M for more Hybrid Platforms

Research

- how to incorporate ZKPs
- how to incorporate micropayments

Business Plan: Pretty Good Apps

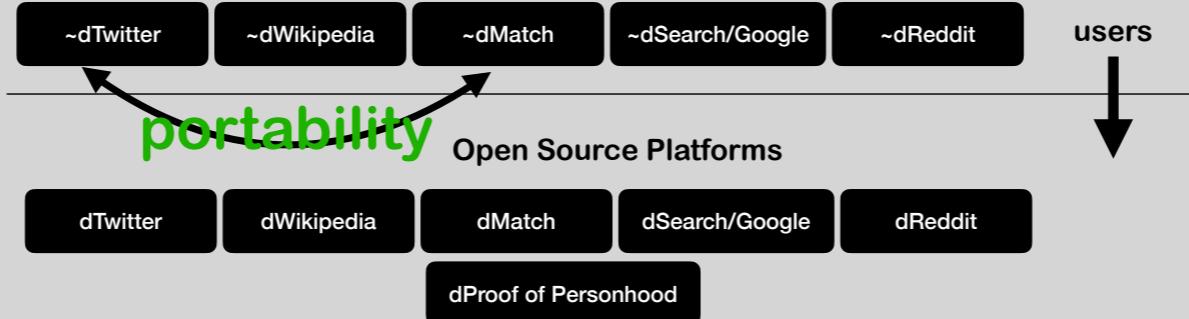
open source Concept Graph and Grapevine as GNU Affero General Public License v3.0

open source communities can use
companies can use, but cannot incorporate into proprietary software

UNLESS I grant license to do so

Proprietary Platforms - I create or lease license

EVIL



Business Plan: Pretty Good Apps

hybrid centralized / decentralized platforms
e.g. ~dWikipedia

- centralized servers and company, similar to existing companies
- database is organized via concept graph
- users have the **OPTION** to export all data
- **PORTABILITY** and **INTEROPERABILITY** with other apps is value proposition
- combined with **EASE OF USE** (same UI as existing)

WHAT IF PGA BECOMES EVIL? LIKE GOOGLE?
ASSUME THIS WILL HAPPEN.

**users will have the option to EXIT ENTIRELY to the DECENTRALIZED
UNDERGROUND**

Business Plan: Pretty Good Apps

choose a web2.0 platform to replace: Twitter, dating app, wikipedia, etc.
license CG and Grapevine to external team

- exclusive
- restricted to platform (e.g. dating app)
- free for short time (1-3 years?)
- PGA gets non-controlling stake in company (10%?) - enough to benefit from success, enough to stay abreast of progress, not enough to get in the way
- after initial period, option to license indefinitely at low cost (cash/year or additional equity); still restricted, but non-exclusive

Business Plan: Pretty Good Apps

rebuild the Concept Graph 0.1 - electron app and server-side

test the CG by conceptualizing of lots of things; when this app proves to be useful to some disinterested 3rd party team (JSON-LD, Verifiable Credentials, etc), then ready to move on to the next step:

rebuild the Grapevine - electron app

open source CG and Grapevine

kick off the Decentralized Underground (for orange pilled)

- launch Proof of Personhood**
- ensure interoperability of disparate platforms: IPFS with GUN, one DID with another, etc**

LAUNCH HYBRID PLATFORMS: bridge for the normies to the dUnderground

use this to fund incorporation of:

BTC/LN

ZKPs for enhanced privacy within the dUnderground

Business Plan: Pretty Good Apps

rebuild the Concept Graph 0.1 - electron app and server-side

test the CG by conceptualizing, then utilizing wherever possible:

- canonical examples, e.g. animals
- Verifiable Credentials
- JSON-LD
- synonym
- the Grapevine
- the Concept Graph - use this to build CG 0.2
- dWikipedia
- dTwitter
- dDating App

rebuild the Grapevine - electron app

open source CG and Grapevine

build Proof of Personhood

Evolution of the Concept Graph over time

development of the Grapevine

- users > developers

development of the Concept Graph

- started as a tool to create new rating templates
- added methods to calculate average scores
- added the ability to manage related concepts (users, product listings, posts, etc)
- ultimately expanded into the Loki Principle

I'd like to make a small digression on the evolution of the CG in my head over the past several years.

I set out many years ago to understand how a decentralized ratings and reputation system ought to work. I had several ideas about how to leave ratings and how to calculate composite scores. Things like: users should have more control over how the ratings and scoring systems work.

EXCISE THIS PART:

trust should be transitive.

it's not trust, it's influence, a weighting, we are trying to calculate. Influence should be a function of average trust score as well as a function of how confident you are in the score, based on how much information you have. Things like: influence should not scale with number of ratings.

GNU Affero General Public License v3.0

I invite input from the cypherpunk community on the strategic use of this license

If this isn't the way, what would be a better way?

Where am I now?
as of x mar 2022

about to rebuild Concept Graph app

Why? reworking UI — property graph — JSON Schema

Qs: alternative to JSON Schema?

need users to edit without breaking the graph

Existing tools: JSON Schema Forms. problem: a concept is not a form.

Contact me if:

you have ideas on how to use CG with YOUR project

you're working with Blue Sky

you are a former Big Tech CEO

you are Jack Dorsey

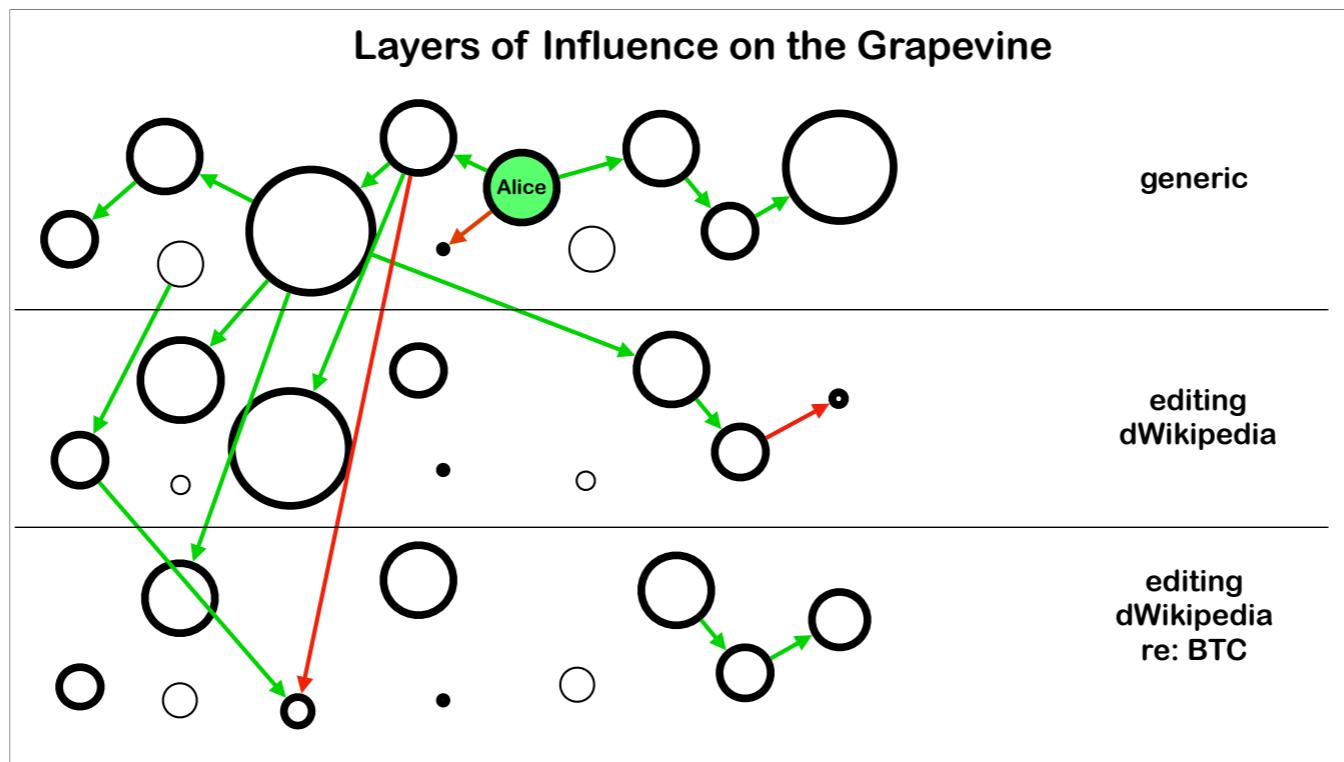
Positions:

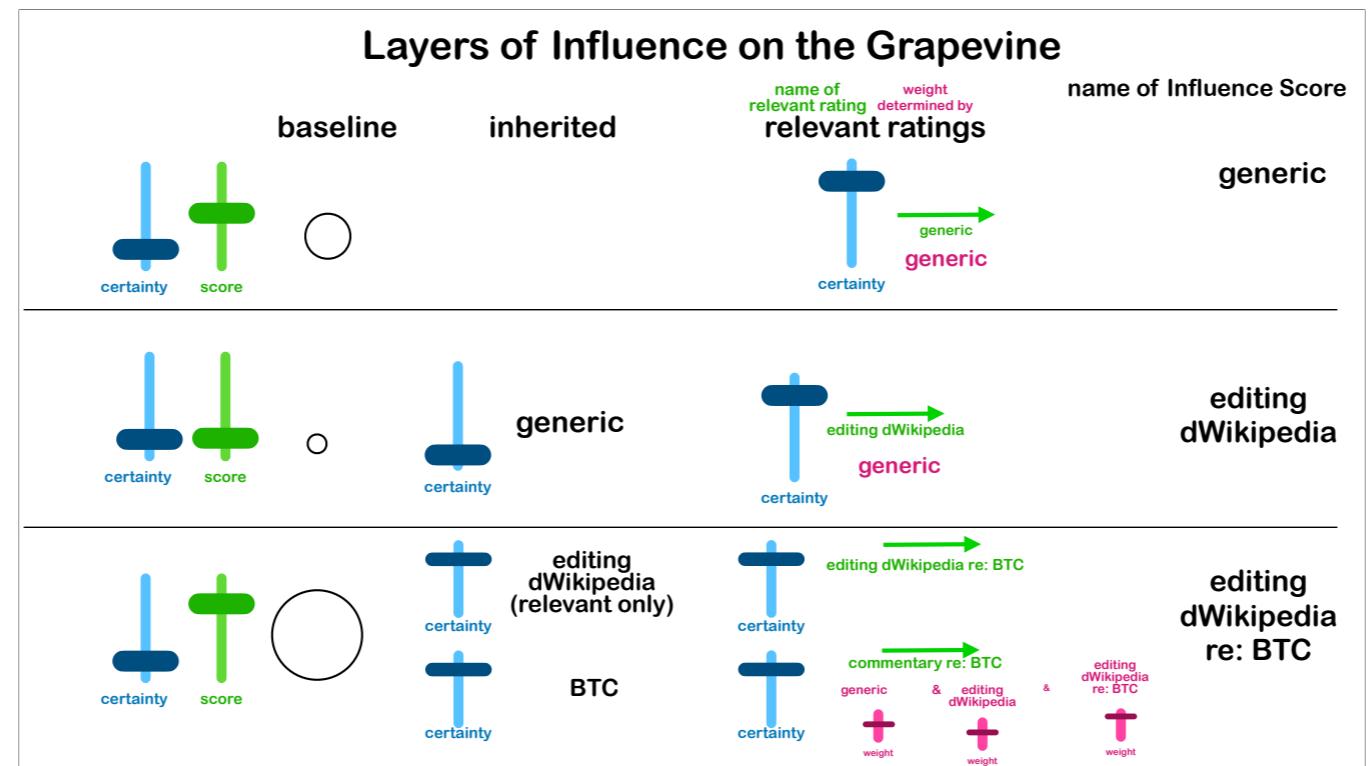
CTO

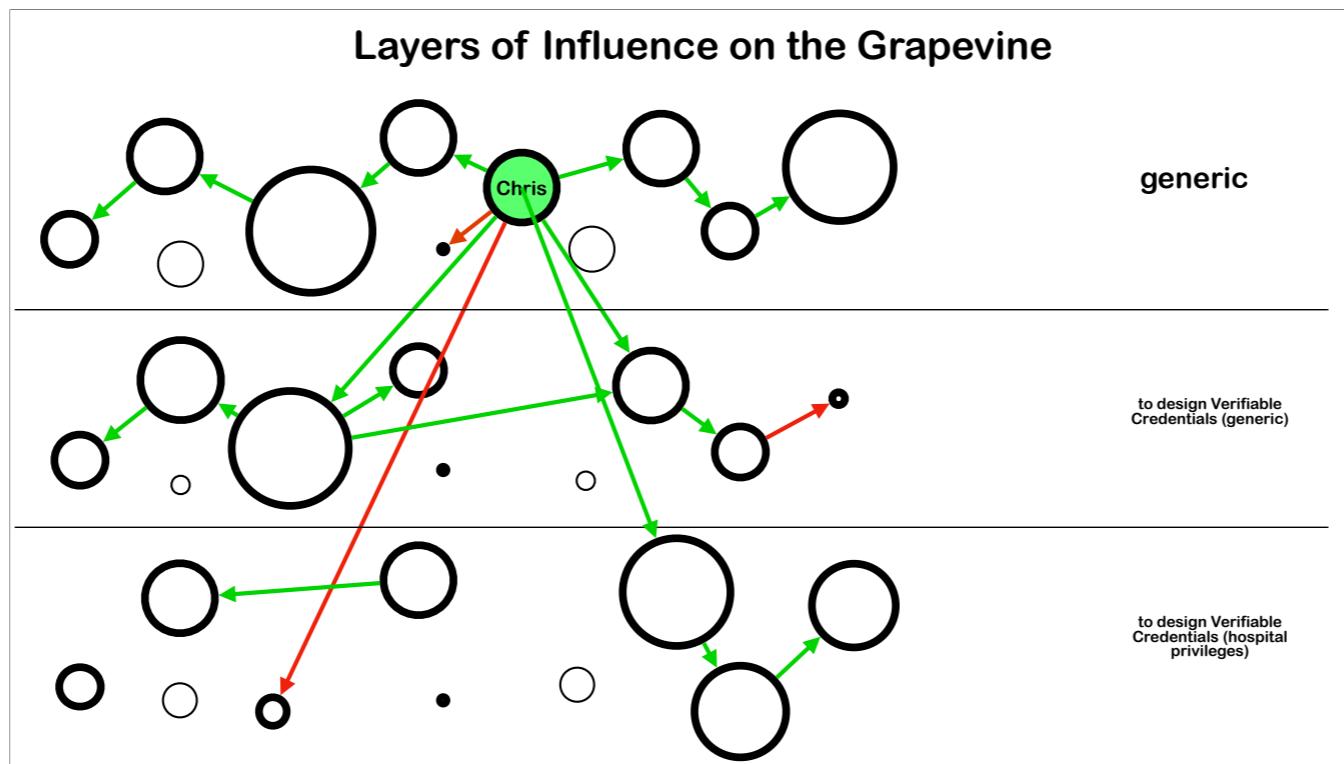
Project Lead: Concept Graph

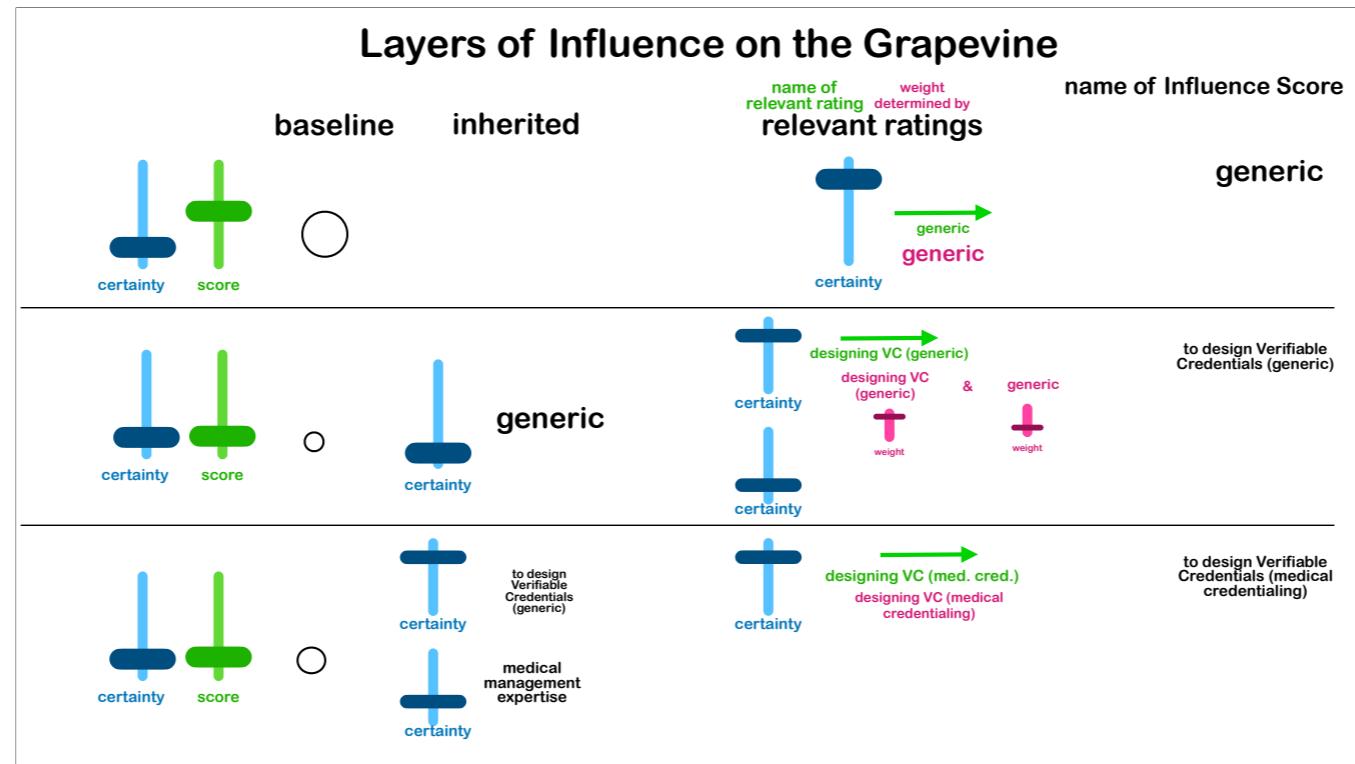
Project Lead: Grapevine

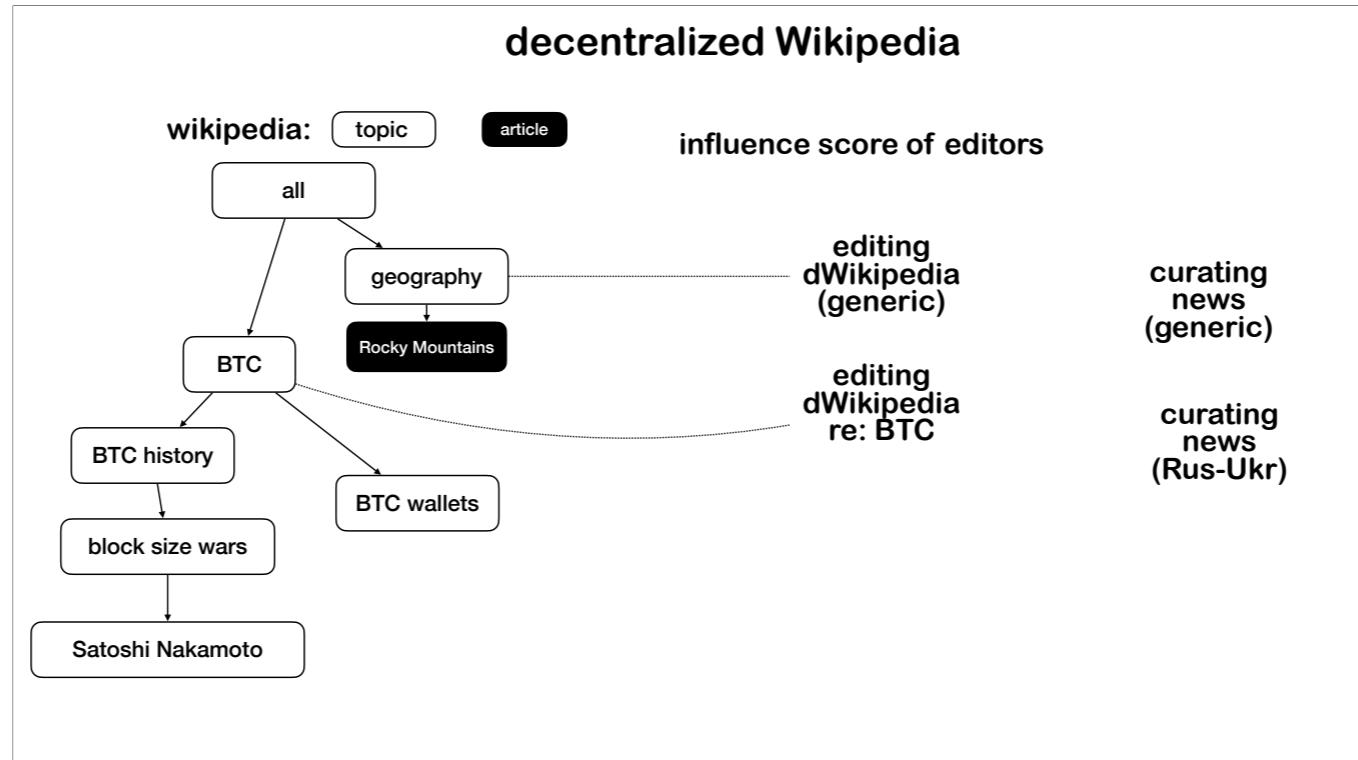
**I AM ASKING: roadmap to be vetted by community
appropriate to use this license?**











**If you are working on dWeb
What does this mean for Your Project?**

Any project can be “conceptualized” by the Concept Graph

frustrated with design decisions?

too many options?

different communities of users may want different designs?

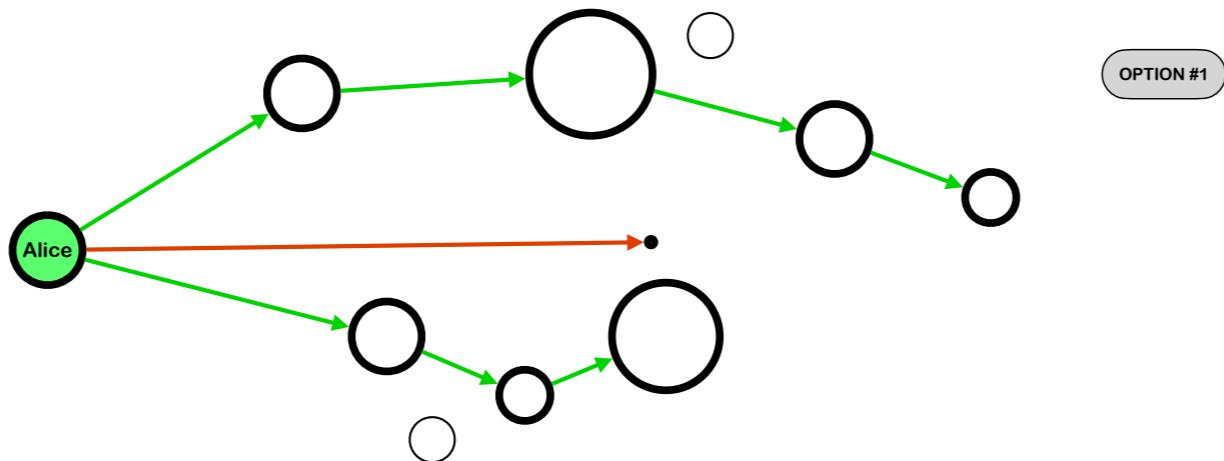
are there any parts of your project that you would like to farm out to third parties?

rating systems

ID systems

Proof of Personhood

able to adjust rapidly to attacks



Inevitable because it's necessary and

decentralized search

Interested in X?

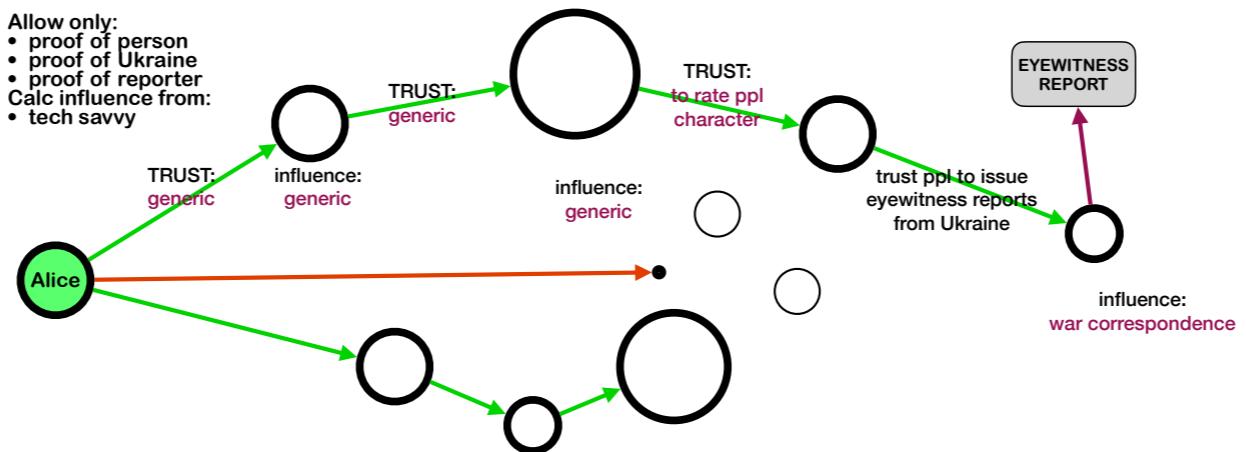
old way: "I'm going to google X."

new way: "What does my Grapevine tell me about X?"

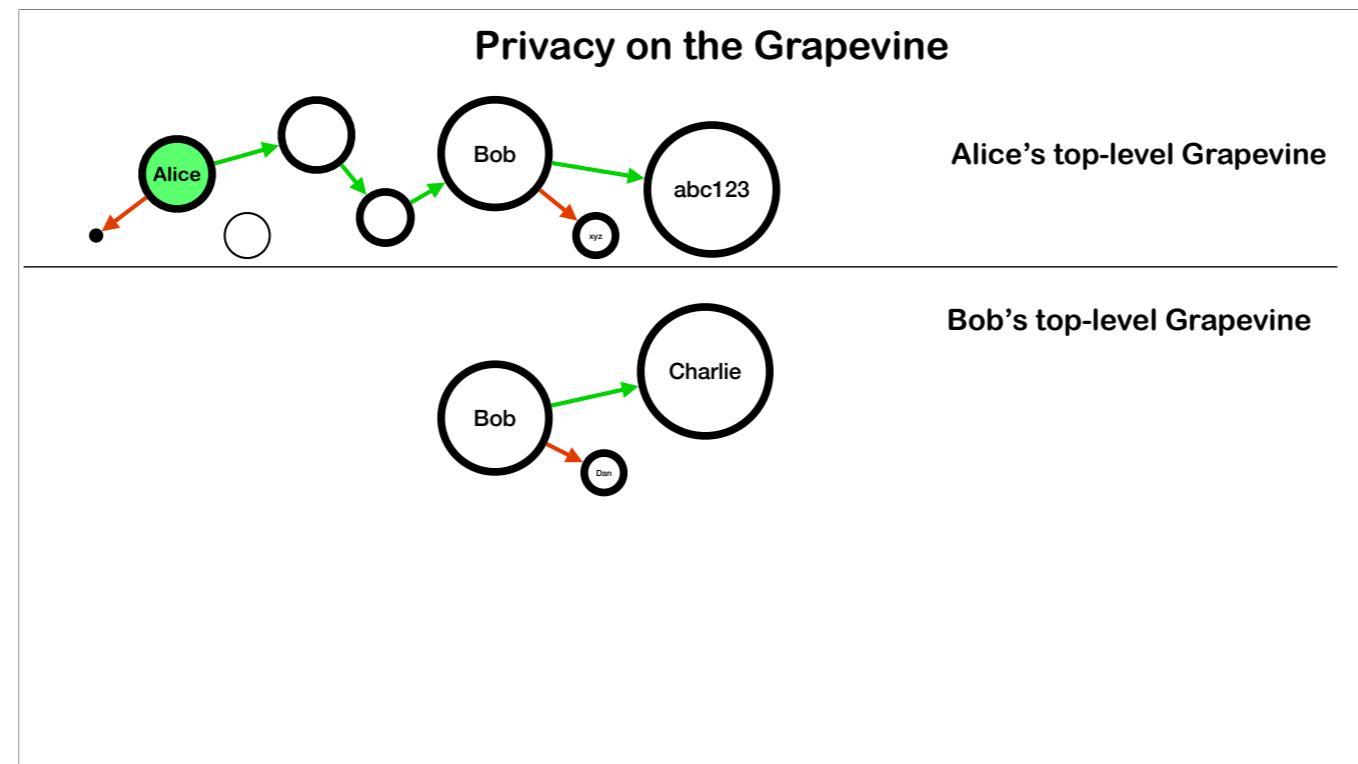
Best use cases: facts rather than subjective opinion (but can be used for both)

Ex: "Show me eyewitness reports from Ukraine."

Allow only:
• proof of person
• proof of Ukraine
• proof of reporter
Calc influence from:
• tech savvy



Inevitable because it's necessary and



Sample app: subReddit

Problem: allow users to create subReddits

```
{
  "peerID": "0ed10nZK8PsPYw5FRhNN2Fj5zmZK6e68805TpJYUuNL7",
  "handle": "",
  "name": "OpenBazaar Store",
  "location": "",
  "about": "",
  "shortDescription": "The official OpenBazaar store run by the development team.  
All sales from this store go into the OpenBazaar project fund.  
Validate this store by comparing it to the link on the official subreddit at https://www.reddit.com/r/OpenBazaar"}
  },
  "shortDescription": "The official OpenBazaar store run by the development team.",
  "nsfw": false,
  "vendor": true,
  "moderate": false,
  "contributors": [
    {
      "website": "openbazaar.org",
      "email": "project@openbazaar.org",
      "name": "",
      "social": [
        {
          "type": "Twitter",
          "username": "openbazaar",
          "proof": ""
        }
      ]
    }
  ],
  "colors": [
    "#FFFFFF",
    "#ECECFF",
    "#28A023",
    "#28A023",
    "#28A023"
  ],
  "avataHashes": [
    "zb2rhrLmRq0139K9z2018pwTjd7SmnqNznb1BzQmXkZk",
    "zb2rhmRo0LNSnRDvF3xTzEx54hdsgXh5mPcVgA019",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "original": "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D"
  ],
  "headerHashes": [
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D",
    "original": "zb2rhgSh0LJqy455Chu7qBmfifr810sgs1TMe6d6ME4D"
  ],
  "stats": {
    "followerCount": 1820,
    "followingCount": 5,
    "listingCount": 1,
    "reviewCount": 119,
    "postCount": 0,
    "averageRating": 4.836364
  },
  "bitcoinPubkey": "0254327ae75f24ba78159dc61aa3a355e15934c6cc8bae06d79dac9e047589e30",
  "tokensAccepted": [
    "2028-04-29T19:27:07.812976172",
    "LTC",
    "BCH",
    "BCH",
    "ZEC",
    "ETH"
  ]
}
```

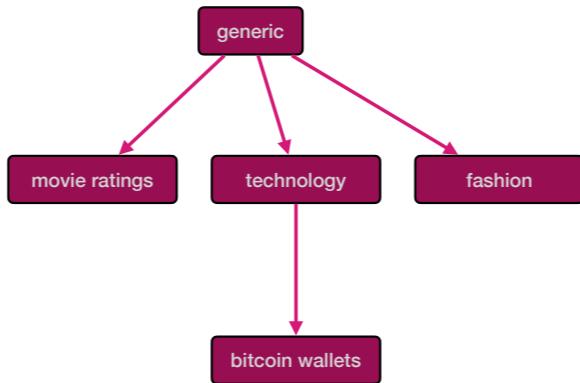
Influence is also Contextual

context is hierarchical

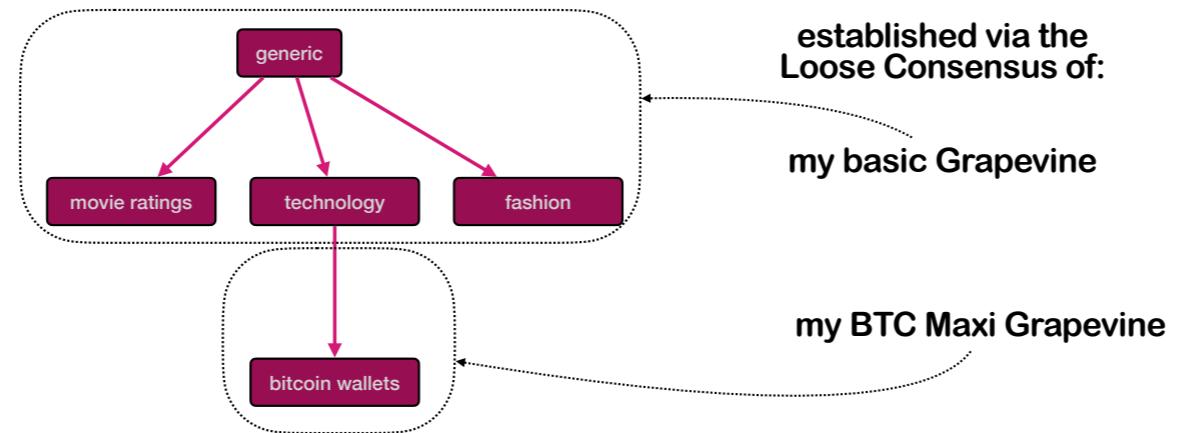
default influence is inherited from higher-level context

context tree is established by loose consensus

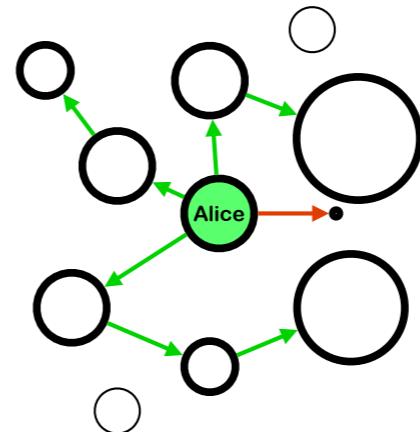
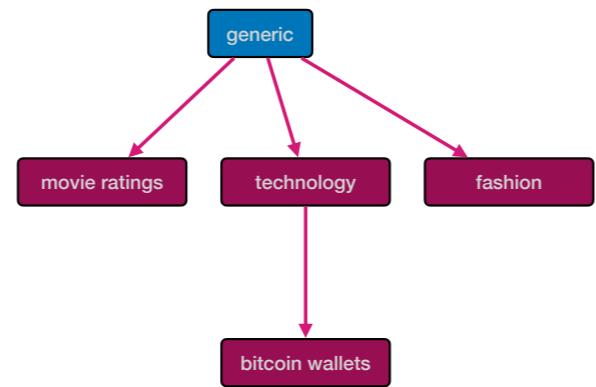
Influence Context Tree



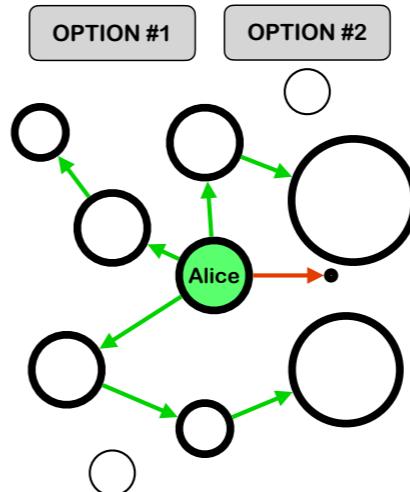
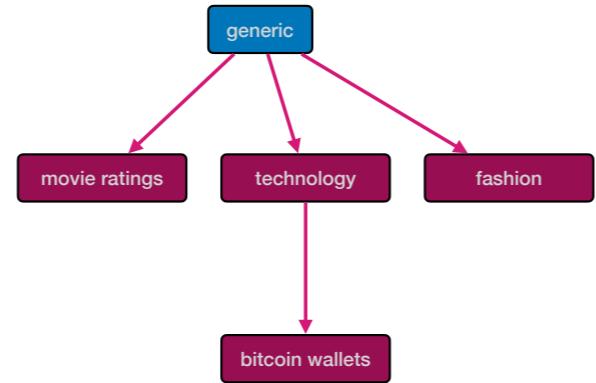
Influence Context Tree



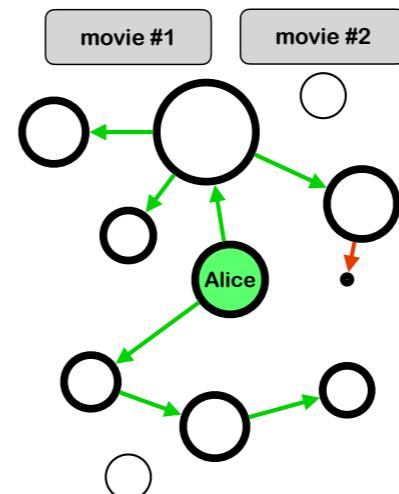
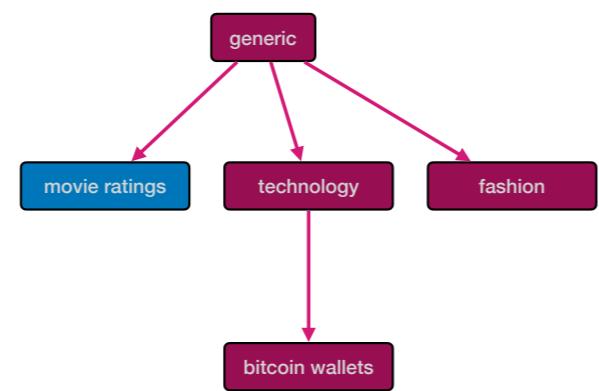
Influence Context Tree



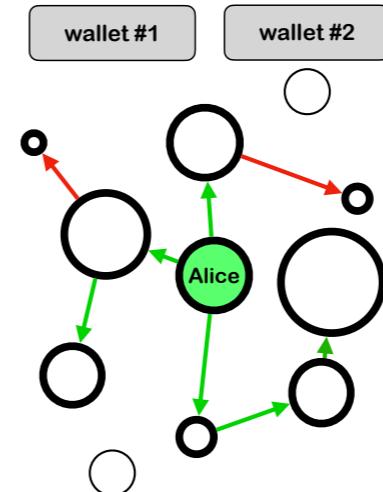
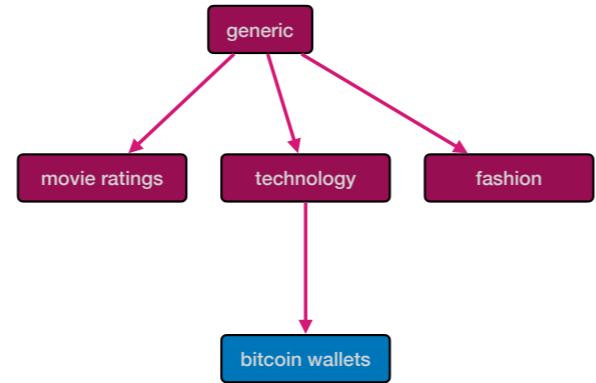
Influence Context Tree



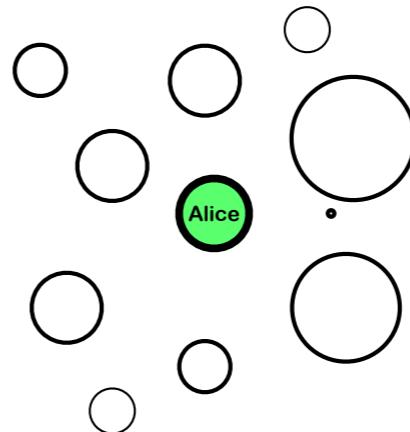
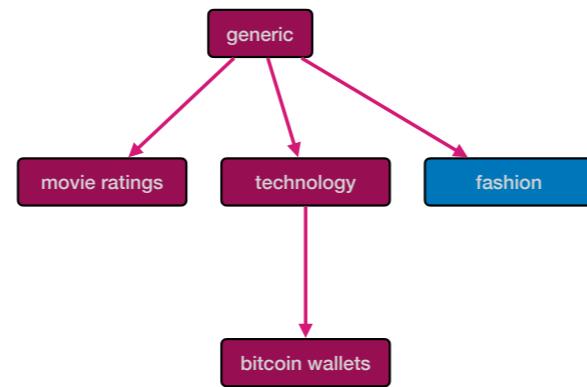
Influence Context Tree



Influence Context Tree

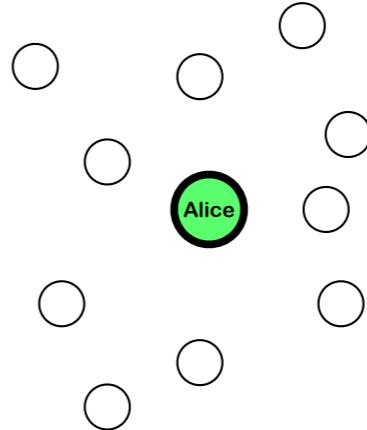
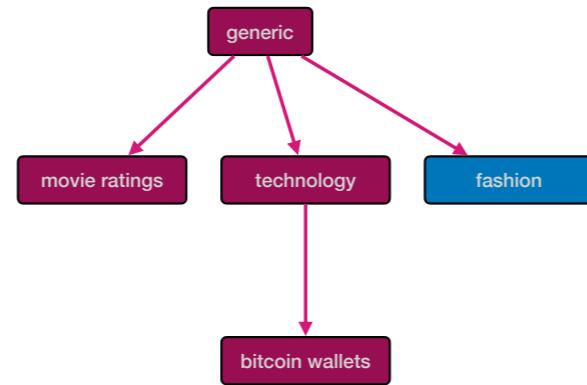


Influence Context Tree



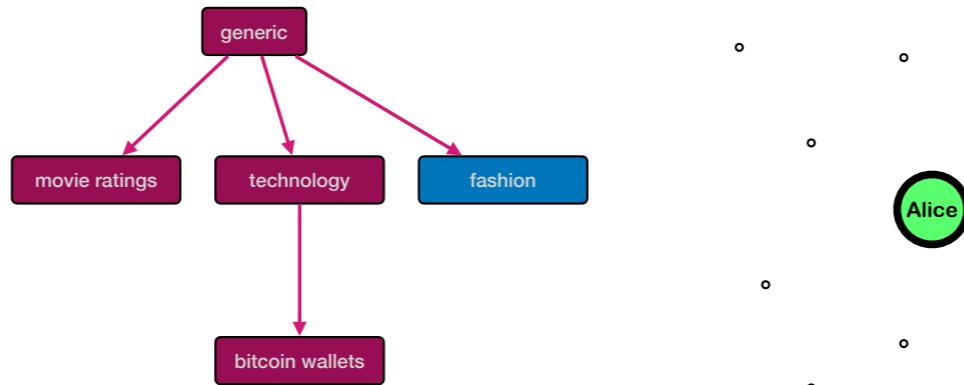
Influence Context Tree

Step 1: set defaults for unknown users



Influence Context Tree

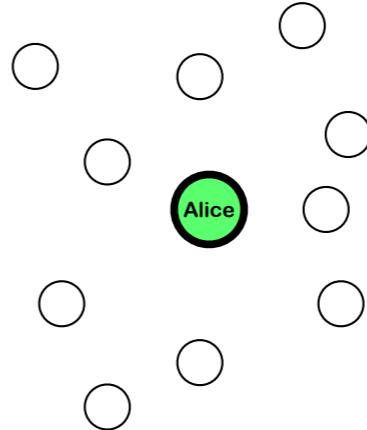
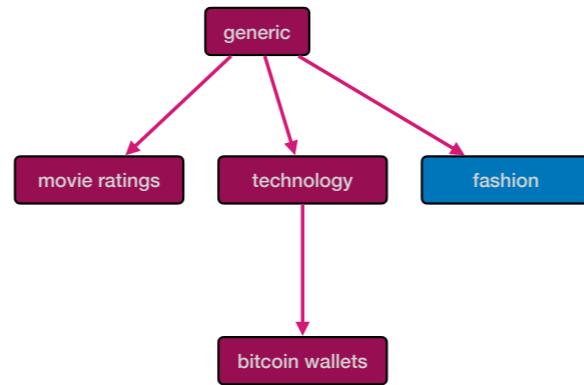
Step 1: set defaults for unknown users



Alice

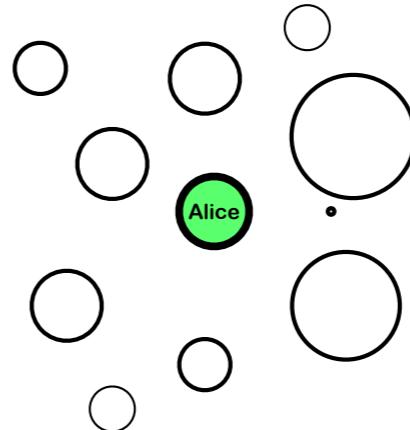
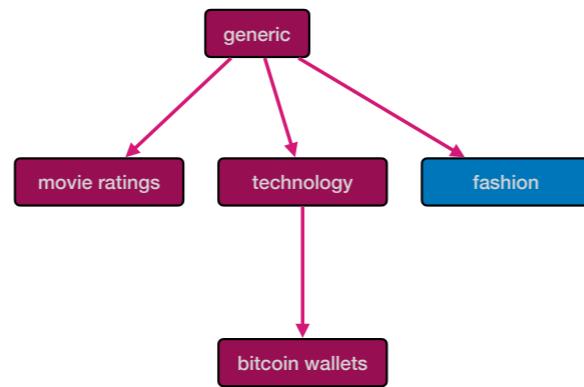
Influence Context Tree

Step 1: set defaults for unknown users



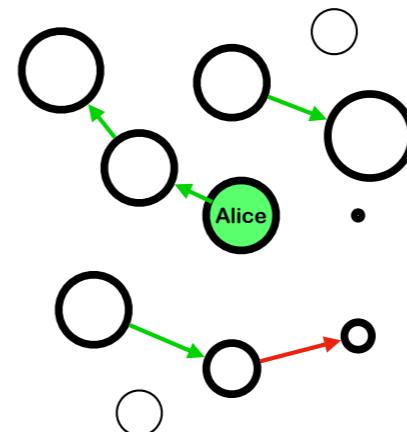
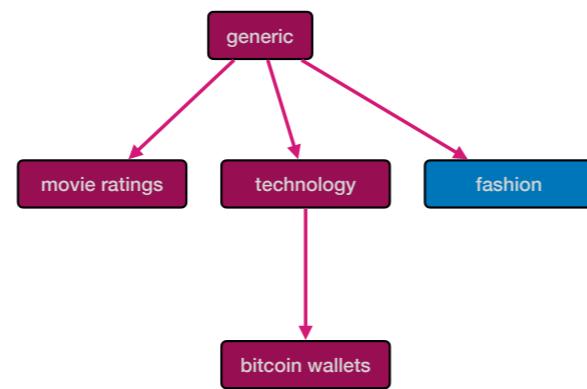
Influence Context Tree

Step 2: inherit score from higher in the hierarchy

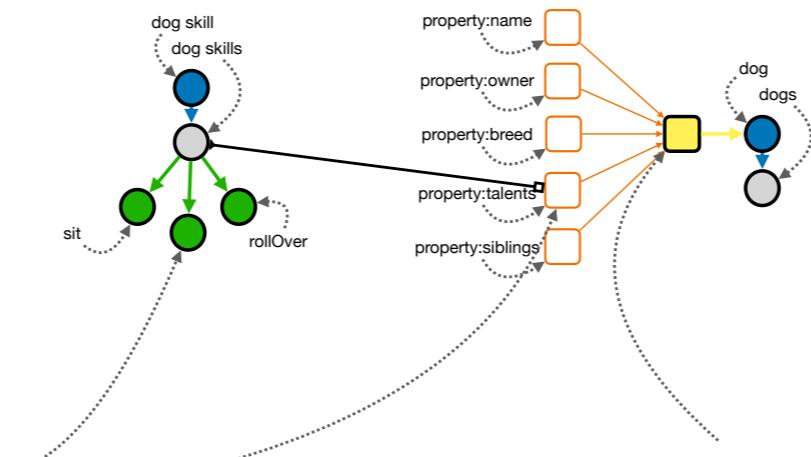


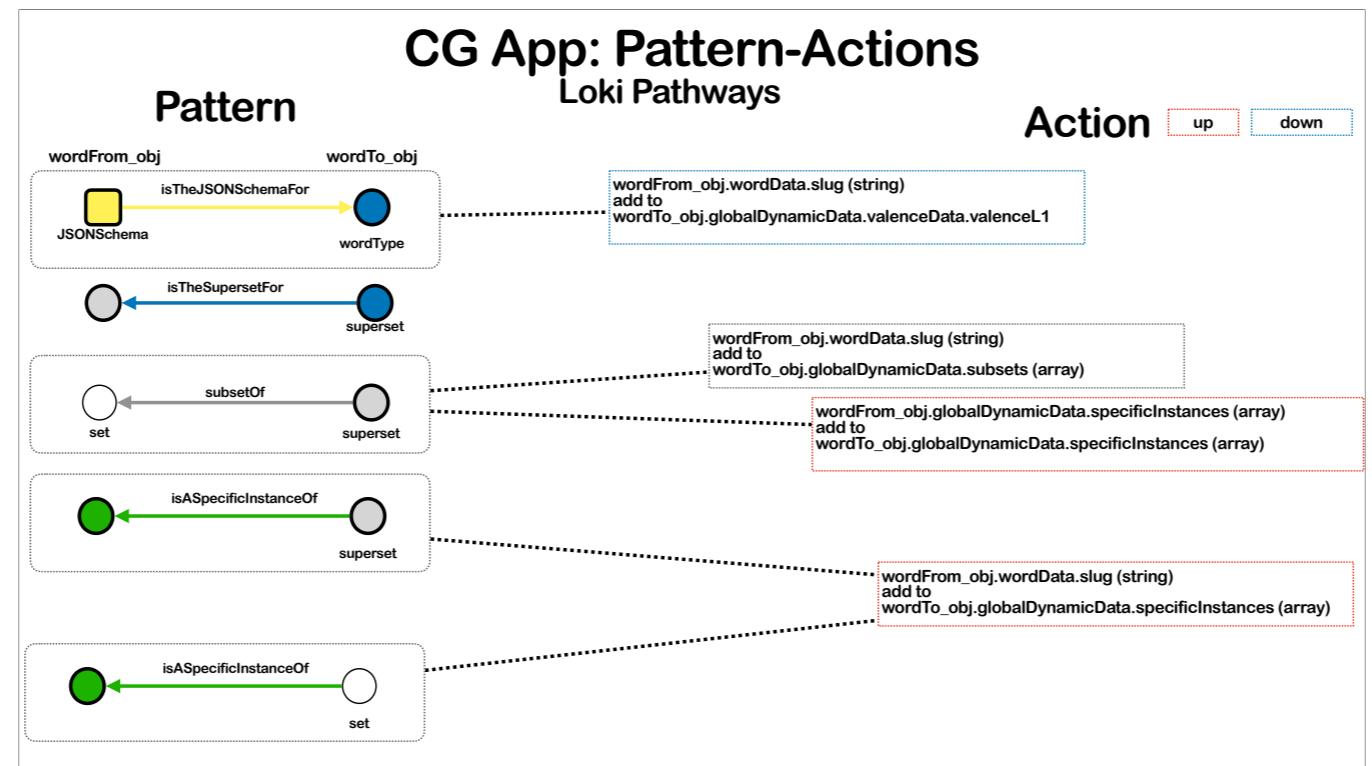
Influence Context Tree

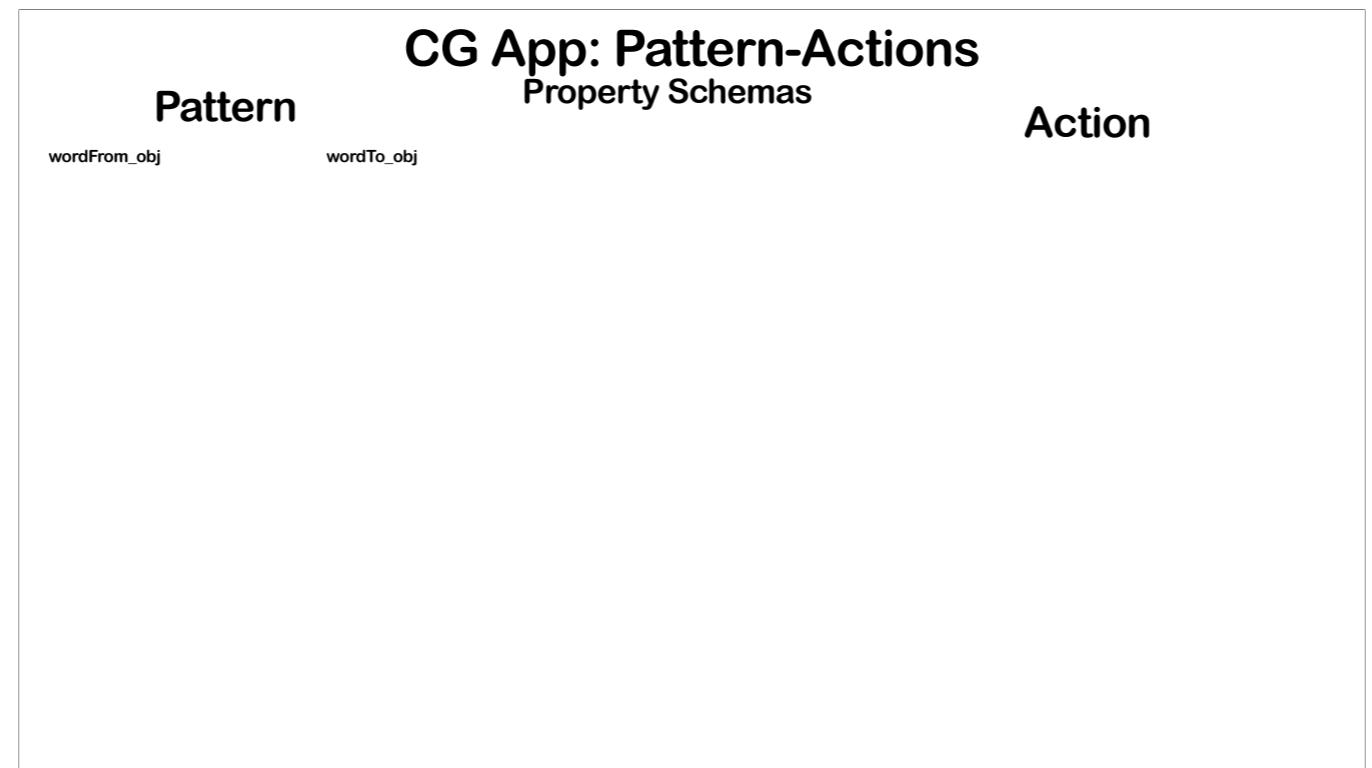
Step 3: input from relevant ratings



Pattern-Actions

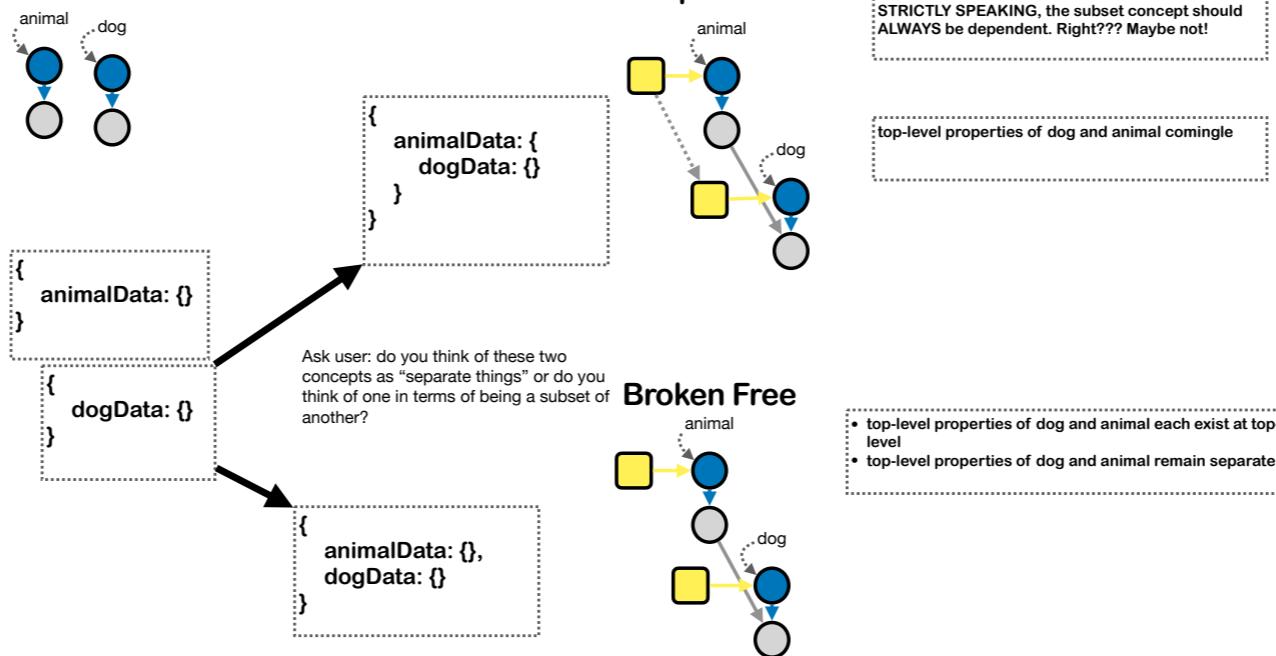




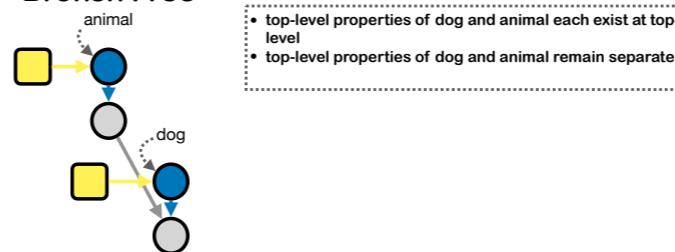


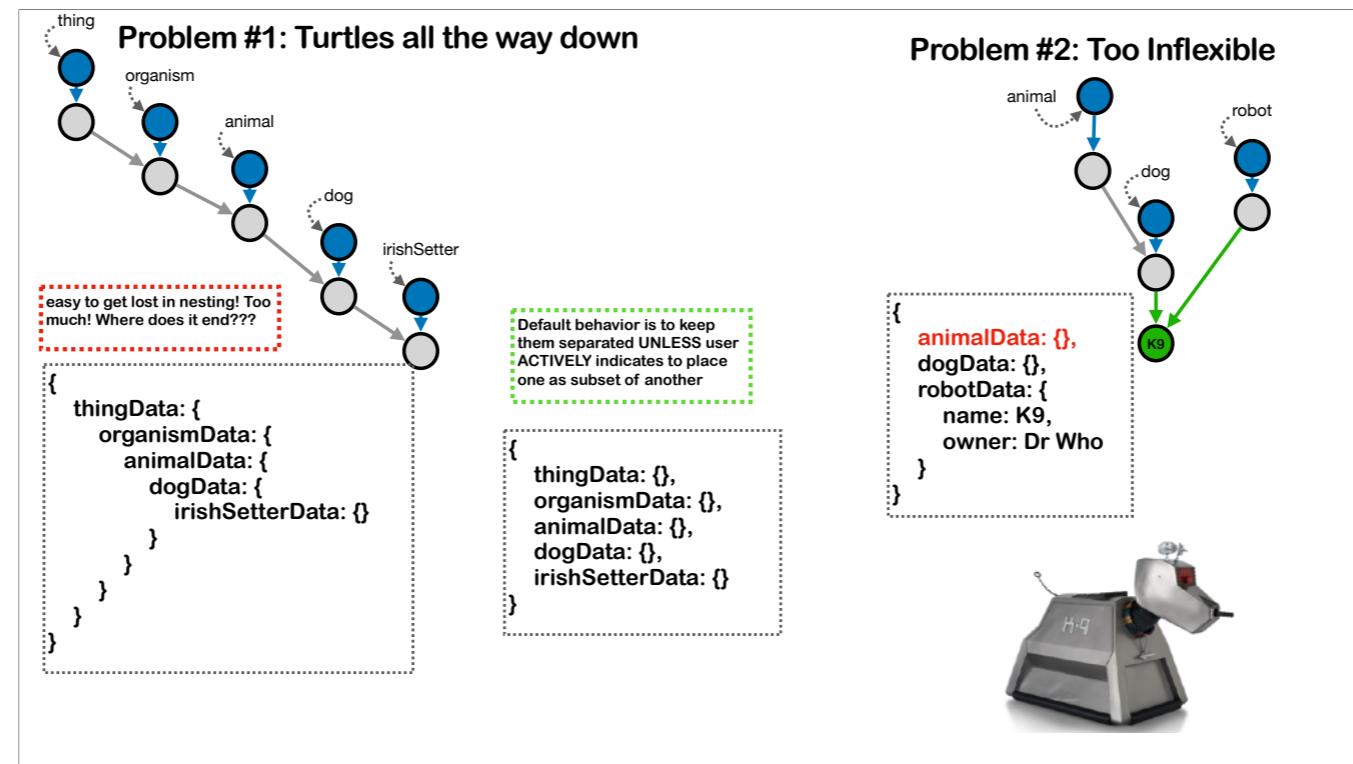
the c2c Subset Dilemma

Dependent

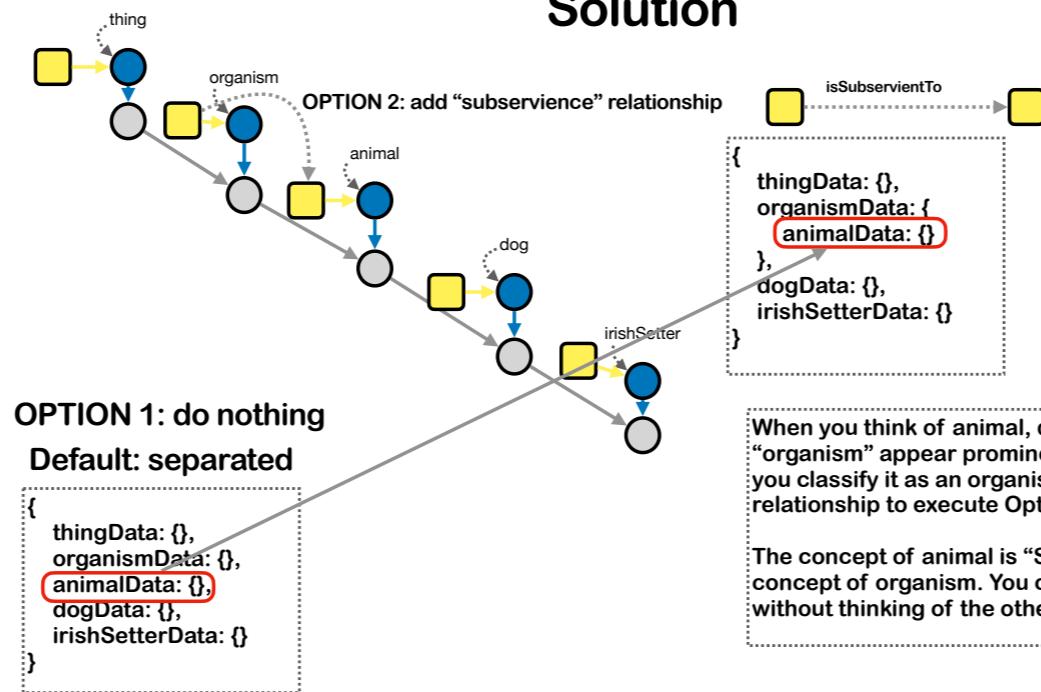


Broken Free

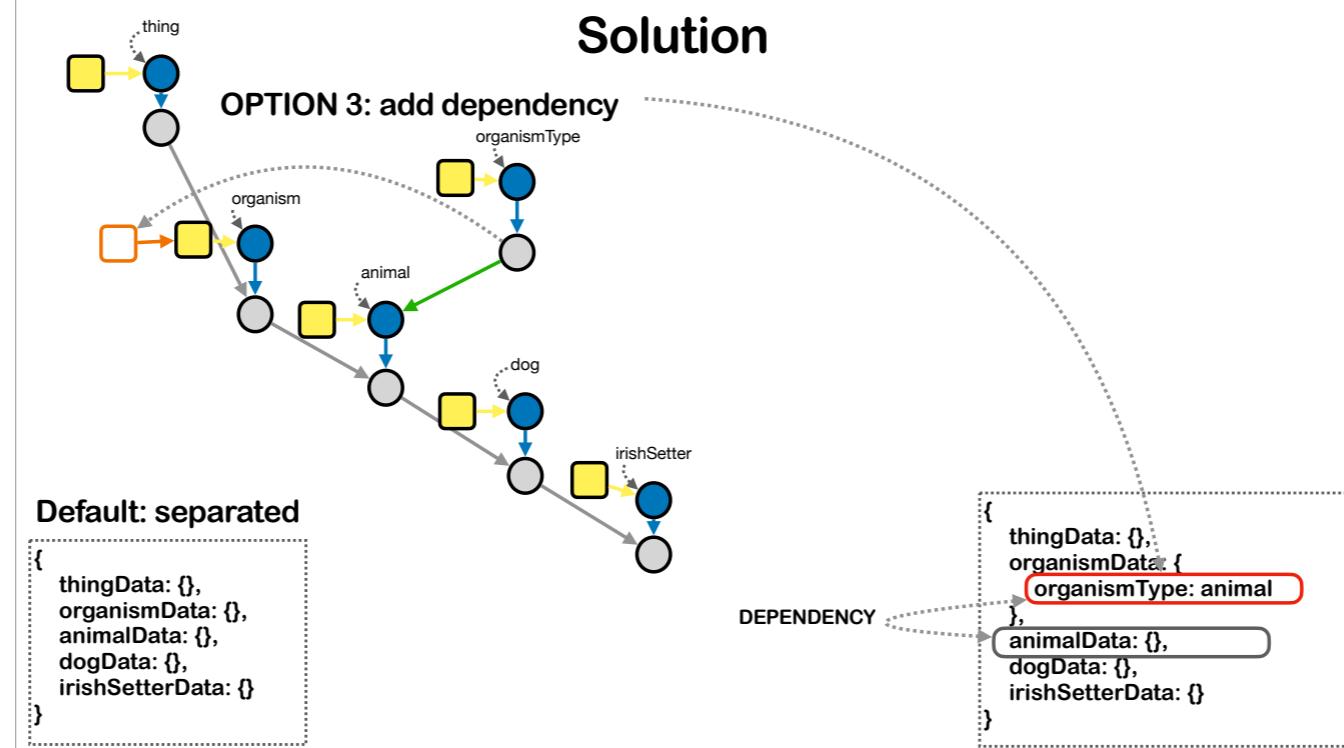




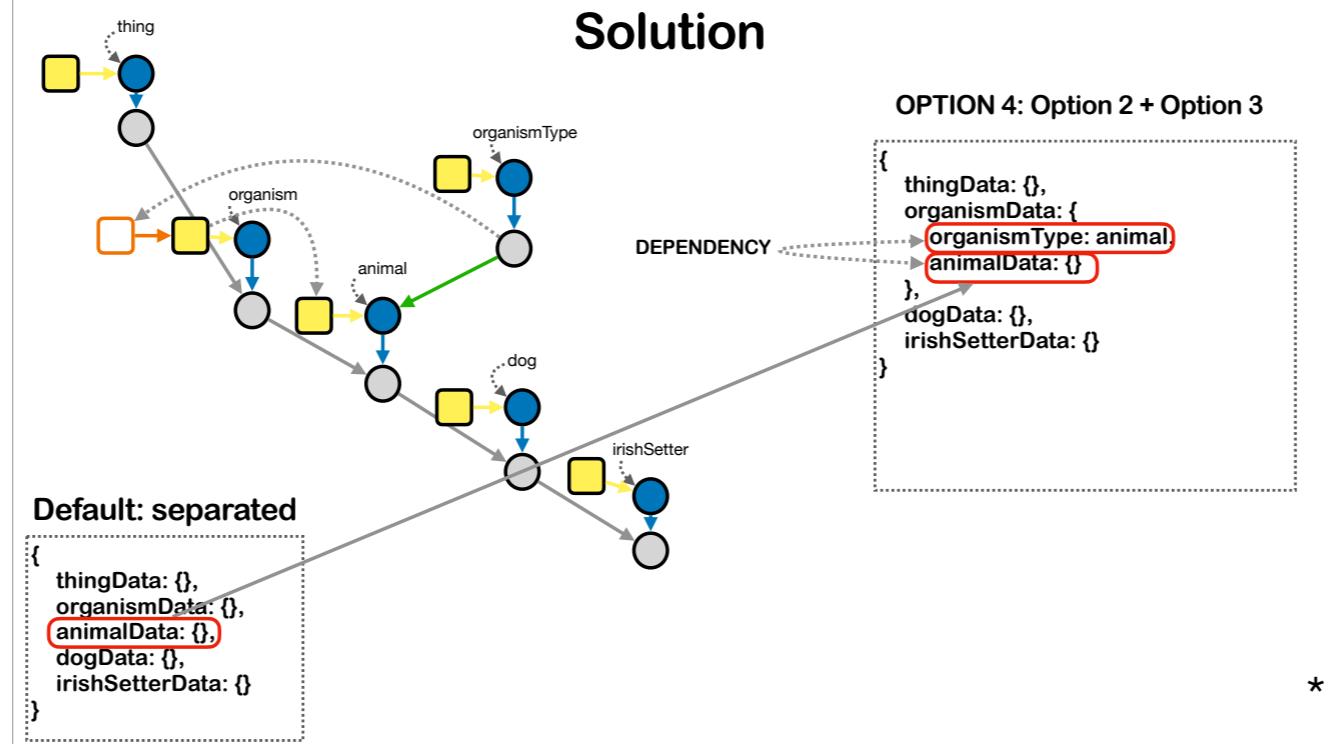
Solution



Solution



Solution



Solution

Question: When implementing OPTION 2, should the user be given the option to identify property in organism with properties in animal? e.g. name of the organism (Spot) == name of the animal (Spot) == the name of the dog (Spot) ? Would this entail deleting the subservient property? Perhaps create a “subservient-delete” relationship from master-property to slave-property?? (call this “PROPERTY IDENTIFICATION”)

if follow OPTION 1: Option 1A == do NOT do property-identification

```
{  
  thingData: {},  
  organismData: {  
    name: Spot  
  },  
  animalData: {  
    name: Spotty  
  },  
  dogData: {  
    name: Spot-man  
  },  
  irishSetterData: {}  
}
```

The diagram illustrates the structure of the objects defined in the code. It shows four nested objects: 'thingData', 'organismData', 'animalData', and 'dogData'. The 'name' property is explicitly defined in each of these nested objects. A red box highlights the 'name' property in both the 'organismData' and 'animalData' objects. Dotted arrows point from these highlighted properties to a text label 'OK to be unequal' located to the right of the 'animalData' object.

OK to be unequal

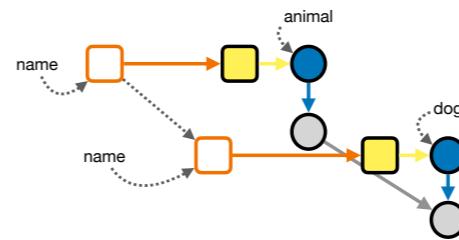
Solution

Question: When implementing OPTION 2, should the user be given the option to identify property in organism with properties in animal? e.g. name of the organism (Spot) == name of the animal (Spot) == the name of the dog (Spot) ? Would this entail deleting the subservient property? Perhaps create a “subservient-delete” relationship from master-property to slave-property?? (call this “PROPERTY IDENTIFICATION”)

if follow OPTION 1: Option 1B == DO property-identification

```
{  
    thingData: {},  
    organismData: {  
        name: Spot  
    },  
    animalData: {  
        name: Spot  
    },  
    dogData: {  
        name: Spot  
    },  
    irishSetterData: {}  
}
```

MUST BE EQUAL

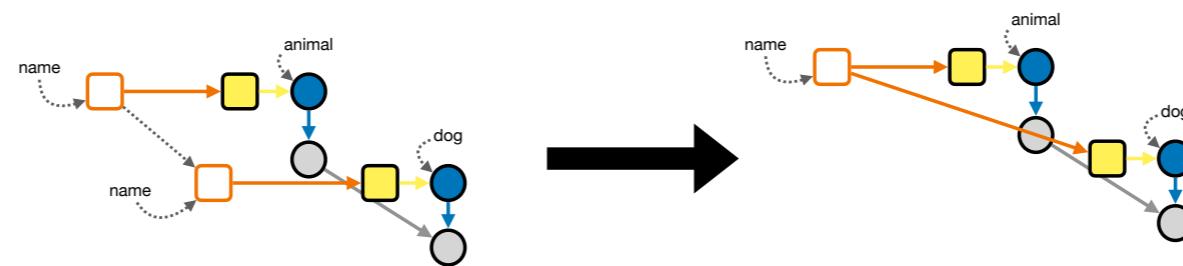


The arrow indicates which one takes precedence over the other!



Solution

A more permanent solution: just delete one of the properties



Solution - UI

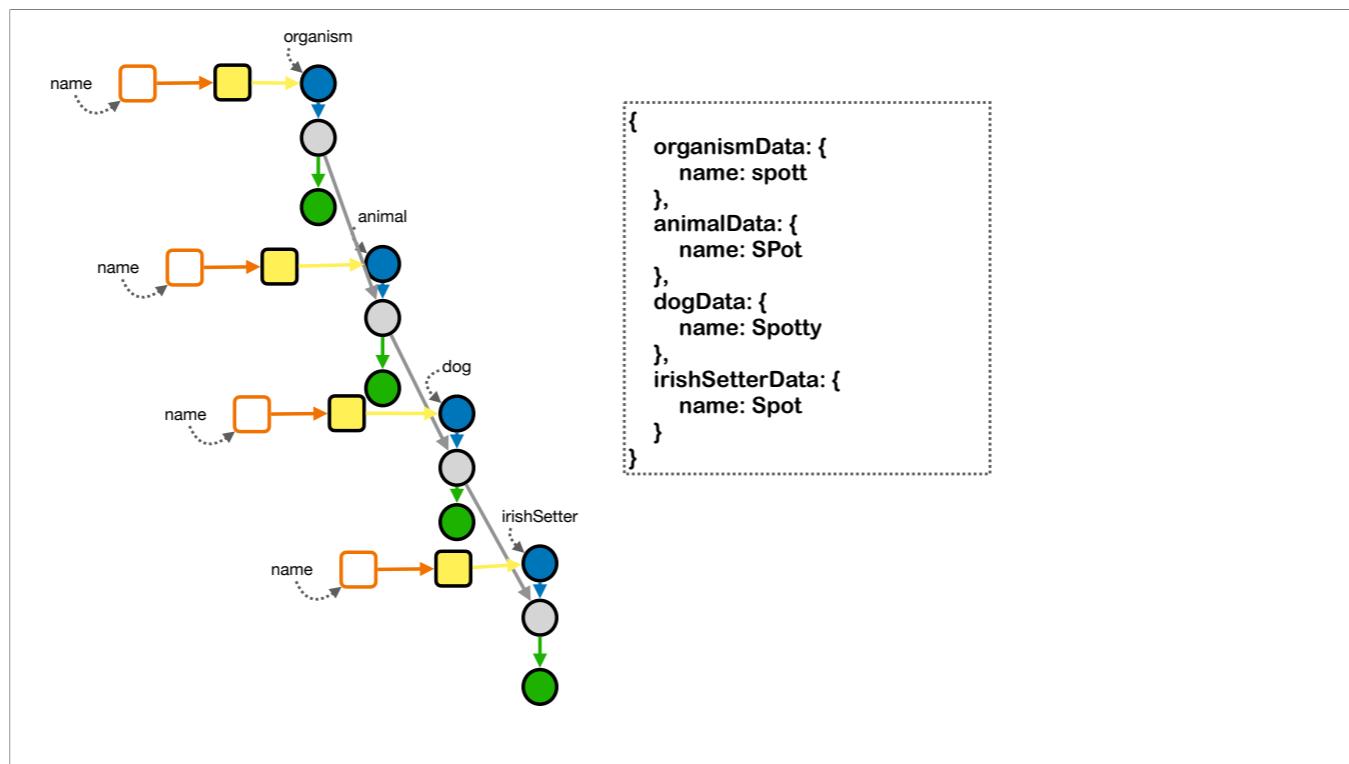
When executing conceptA — isASubsetOf — conceptB

The user is prompted with several subsequent options:
Create a conceptA_Type concept (if one does not already exist)?

Execute “OPTION 2” to make conceptA “SUBSERVIENT” to ConceptB?

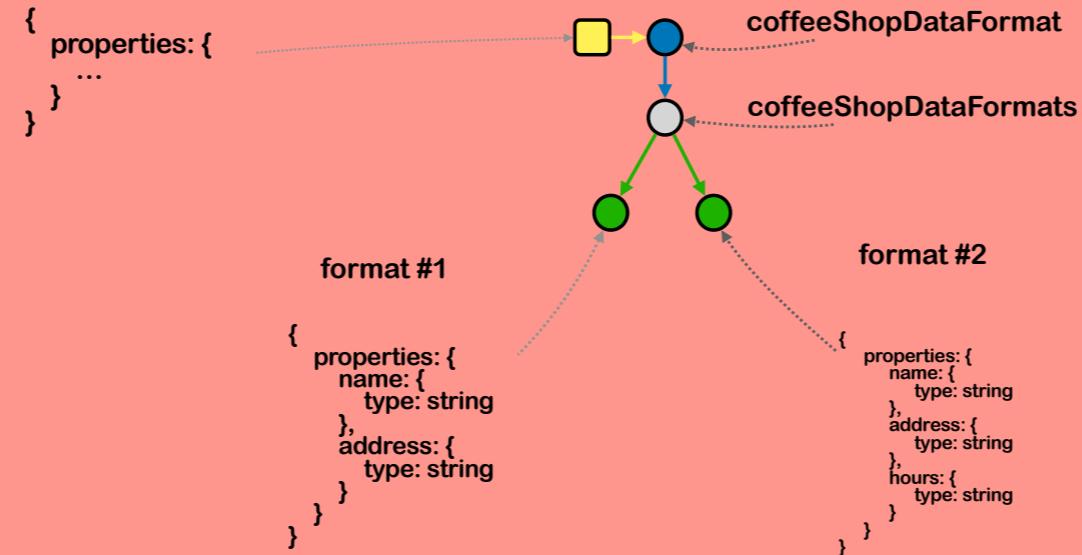
Execute “OPTION 3” to establish DEPENDENCY?

Identify properties of conceptA with properties of conceptB? (top level only
probably? Would it make sense to go deeper than that?)



Relationship between the Concept Graph and the Grapevine

concept: "formatForCoffeeShop"



The GRAPEVINE App

Front end:

- leave ratings
- view Grapevine, with Influence scores

So that's how Alice could view her Grapevine and view and manage calculation of Influence Scores in a contextual manner.

The GRAPEVINE App

Front end:

- leave ratings
- view Grapevine, with Influence scores
- create new ratings and new ways to calculate average scores

The other thing Users will be able to do will be able to create new ratings and new ways to calculate average scores. As before, some of this might happen on more specialized apps, like an app for decentralized Twitter. But it could potentially happen in either app [i.e. we could use either the baseline Grapevine app or a more specialized app to create new ratings.]. What does that mean? Let's take an example.

Sample app: dTwitter

Suppose Alice is using a platform which is a decentralized Twitter.

Sample app: dTwitter

Problem: filter out ETH-giveaway Spam-bots

She notices a problem, which is: bots that are claiming to give away Ethereum or other tokens. She would like to filter these bots out of her feed! How can she do this?

Sample app: dTwitter

Problem: filter out ETH-giveaway Spam-bots

Solution:

- user creates new rating: flag user: ETH Spam Bot
- flag user: ETH Spam Bot Hunter

The solution would be for her to create a new rating.

dTwitter USE CASE: Elimination of “ETH Giveaway!” Spam

Step 1: creation of several new rating templates:

- flag tweet as ETH Giveaway Spam
- flag user as ETH Giveaway Troll
- flag user as Troll Hunter

Step 2: creation of new composite scores to reflect whether users are Trolls or Troll Hunters

Step 3: bundling of all of the above together; public promotion; other users flag the bundle as worthy of adoption

No need for approval by any “core developer” team; execution all from users who are not developers

The Concept Graph App

Why create a Concept Graph?

loose consensus

developer: conceptualize all or part of an app
identify parts that you want fixed and parts that you want users to be able to edit. Compared to relational db, easier to change without breaking things. So import the CG app functionality into your new app.

as a user: make changes as envisioned by the developer
make concept graphs for fun!
use CG to edit the grapevine, build loose consensus

Should any part of my project be conceptualized?

For anyone working on dWeb, I would ask you to consider what it would look like if you were to conceptualize a part of the project you are working on.

Should any part of my project be conceptualized?

Advantages

- Loose Consensus
 - give users greater control over capabilities
 - different solutions for different users or communities
- improved documentation
- interoperability with outside projects
- farm difficult decisions to third parties

For anyone working on dWeb, I would ask you to consider what it would look like if you were to conceptualize a part of the project you are working on.

Should any part of my project be conceptualized?

Advantages

- improved documentation
- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large
components
↑
↓
small
details

Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter

Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large
components
↑
↓
small
details

Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter



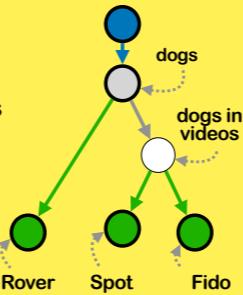
Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

Examples:

- Funny Dog Videos
 - Open Bazaar third party search engines
 - Front End of any app
 - JSON-LD
 - rating system
 - middle-name field for user in a decentralized Twitter



Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

Examples:

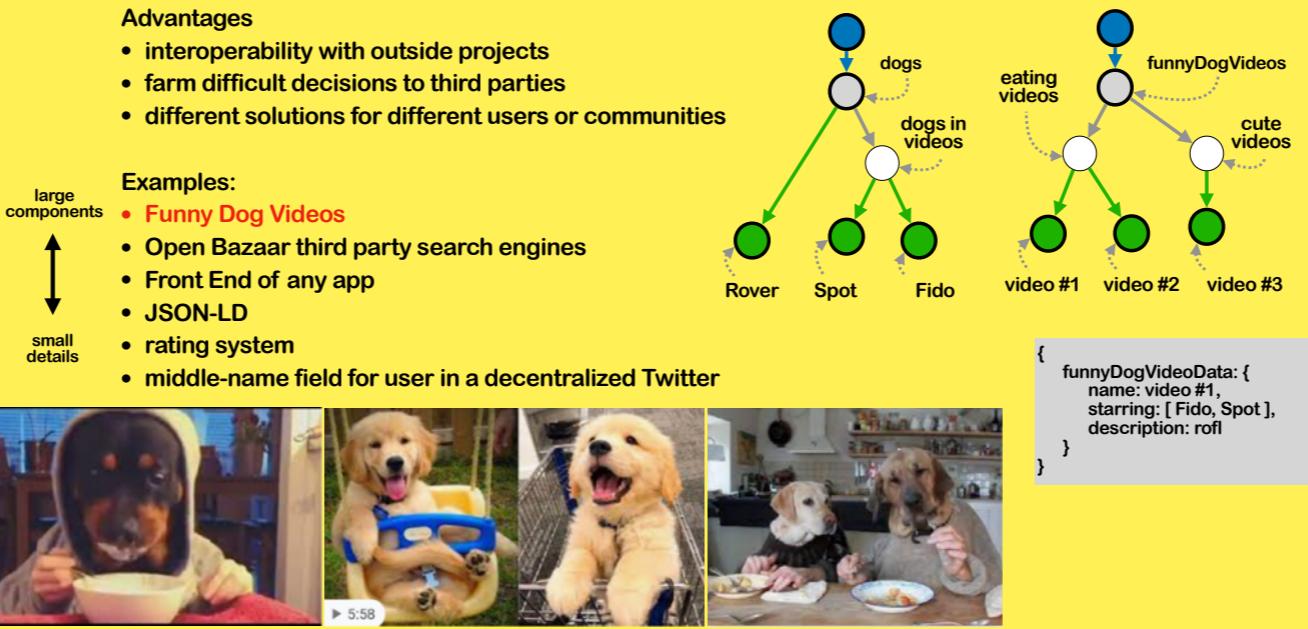
- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter

large components
↔
small details

```
{  
  funnyDogVideoData: {  
    name: video #1,  
    starring: [ Fido, Spot ],  
    description: rofl  
  }  
}
```

Why do it this way? Because this will allow users to rely on their Grapevines to curate their lists of dogs and dog videos. Not only add more videos, but also the Grapevine can change how they are organized.

Should any part of my project be conceptualized?



No need to wait for developers to do all that.

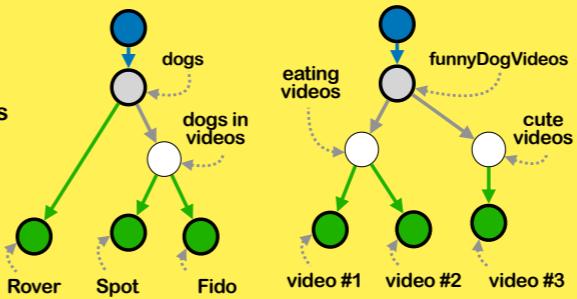
Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

Examples:

- Funny Dog Videos
 - Open Bazaar third party search engines
 - Front End of any app
 - JSON-LD
 - rating system
 - middle-name field for user in a decentralized Twitter



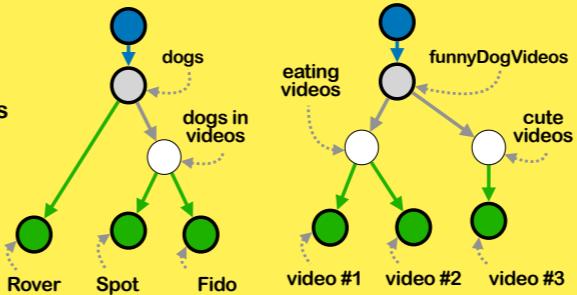
Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

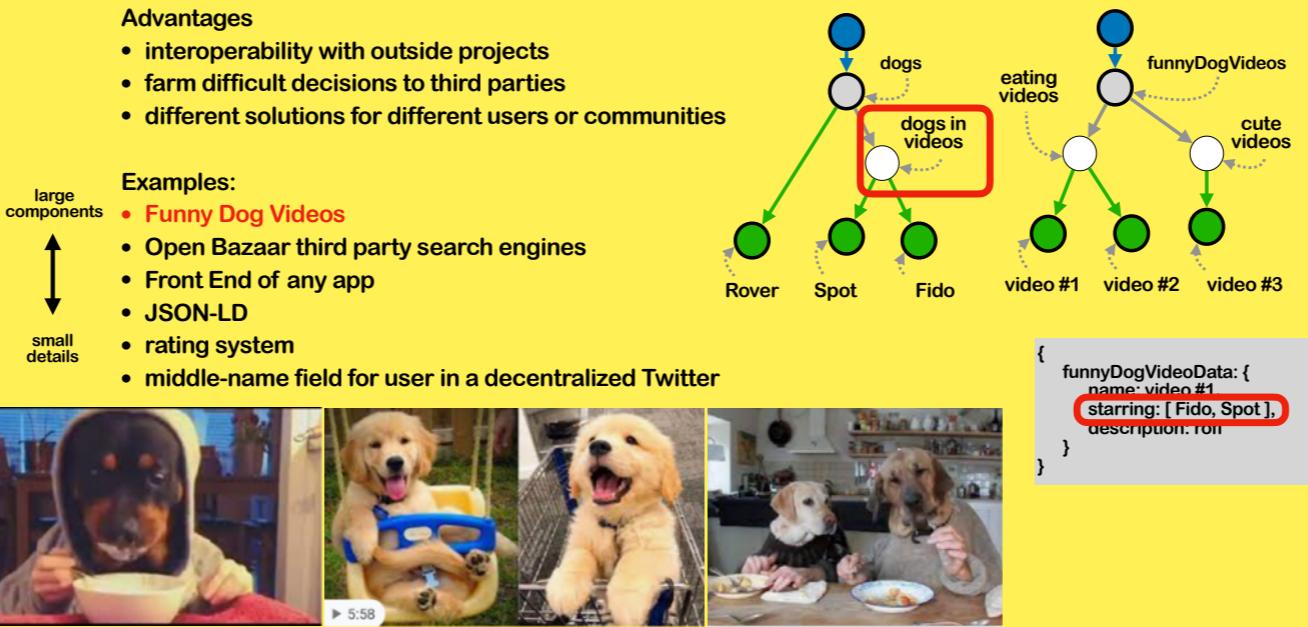
Examples:

- Funny Dog Videos
 - Open Bazaar third party search engines
 - Front End of any app
 - JSON-LD
 - rating system
 - middle-name field for user in a decentralized Twitter



```
{  
  funnyDogVideoData: {  
    name: video #1,  
    starring: [ Fido, Spot ],  
    description: rofl  
  }  
}
```

Should any part of my project be conceptualized?

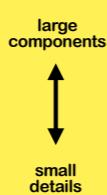


and you would want to relate the two concepts so that the elements of this property are constrained to be limited to members of the set of dogs in videos.

Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities



Examples:

- Funny Dog Videos
- **Open Bazaar third party search engines**
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter

Another example: Open Bazaar. It was designed so that the search engine function was farmed out to third parties. The third parties would scrape the network, maintain a database of vendors and listings, and would deliver search results quickly.

Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

OB1 Search Engine
Alice's Search Engine



Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter

Should any part of my project be conceptualized?

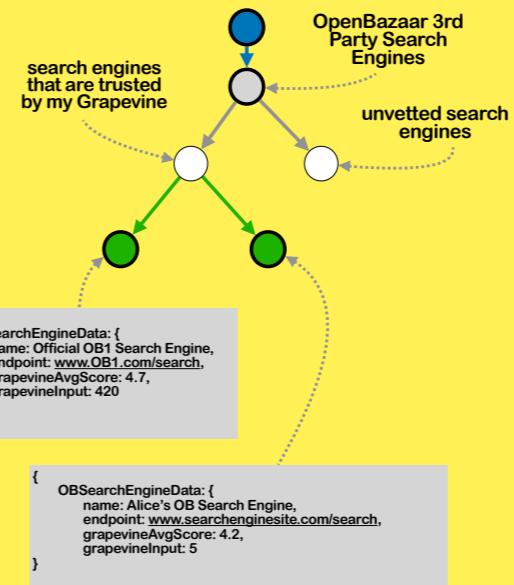
Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large components
↔
small details

Examples:

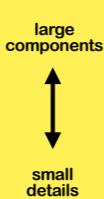
- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter



Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities



Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- **JSON-LD**
- rating system
- middle-name field for user in a decentralized Twitter

Another example: Linked Data.

Should any part of my project be conceptualized?

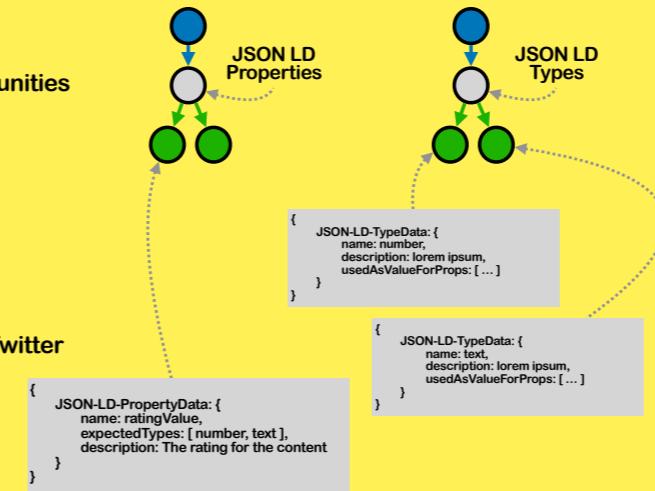
Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large components
↔
small details

Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- **JSON-LD**
- rating system
- middle-name field for user in a decentralized Twitter



Should any part of my project be conceptualized?

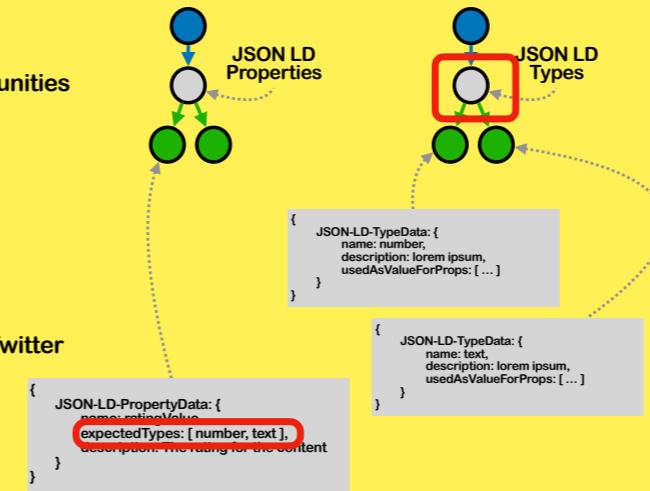
Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large components
↔
small details

Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- **JSON-LD**
- rating system
- middle-name field for user in a decentralized Twitter



Should any part of my project be conceptualized?

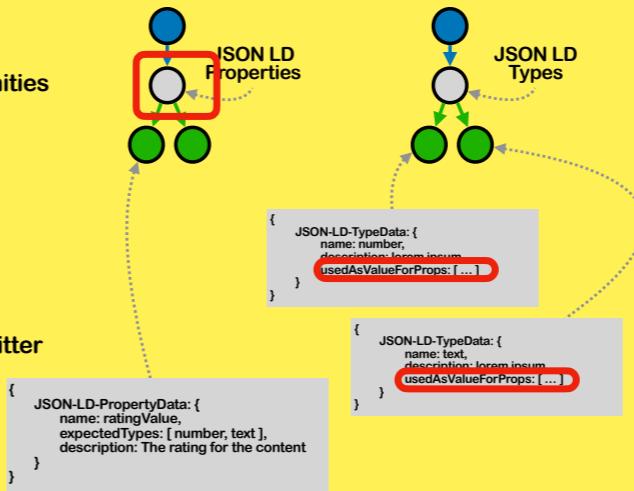
Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large components
↔
small details

Examples:

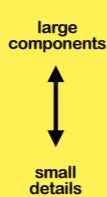
- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- **JSON-LD**
- rating system
- middle-name field for user in a decentralized Twitter



Should any part of my project be conceptualized?

Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities



Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter

Should any part of my project be conceptualized?

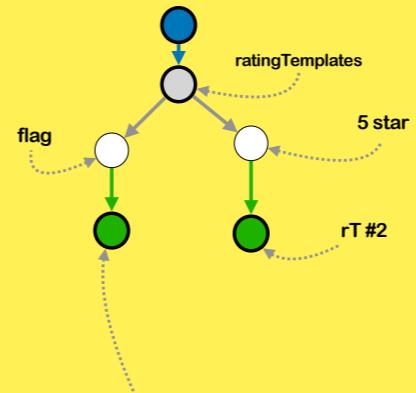
Advantages

- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

large components
↔
small details

Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- **rating system**
- middle-name field for user in a decentralized Twitter

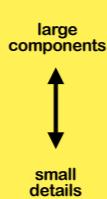


```
{  
  ratingTemplateData: {  
    name: Flag Scammer,  
    description: flag users who are suspected scammers,  
    type: flag,  
    target: decentralized Twitter user  
  }  
}
```

Should any part of my project be conceptualized?

Advantages

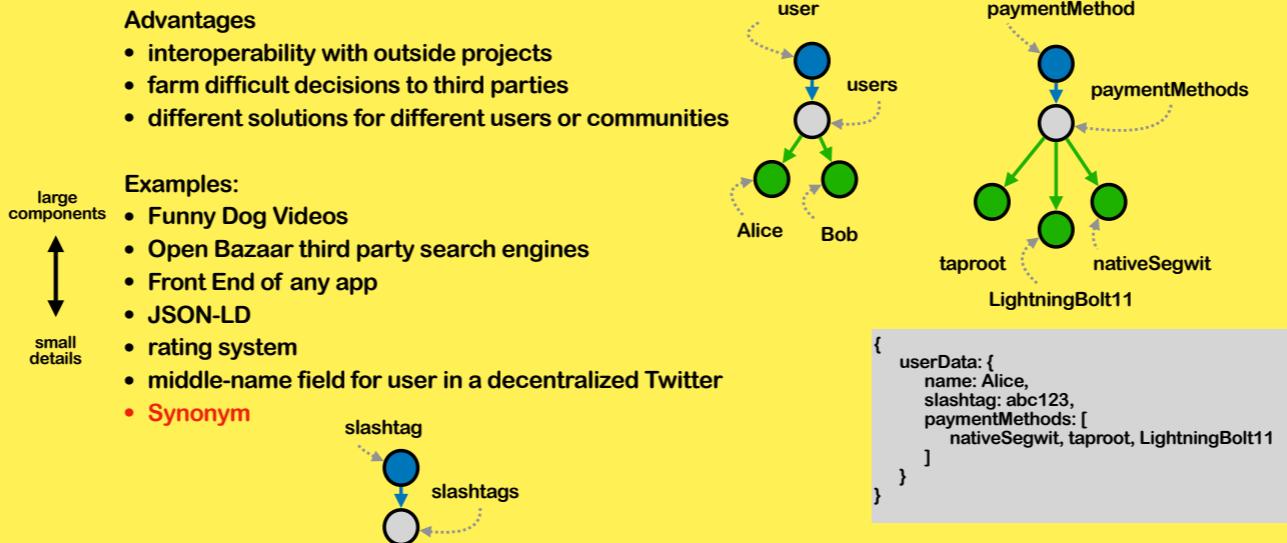
- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities



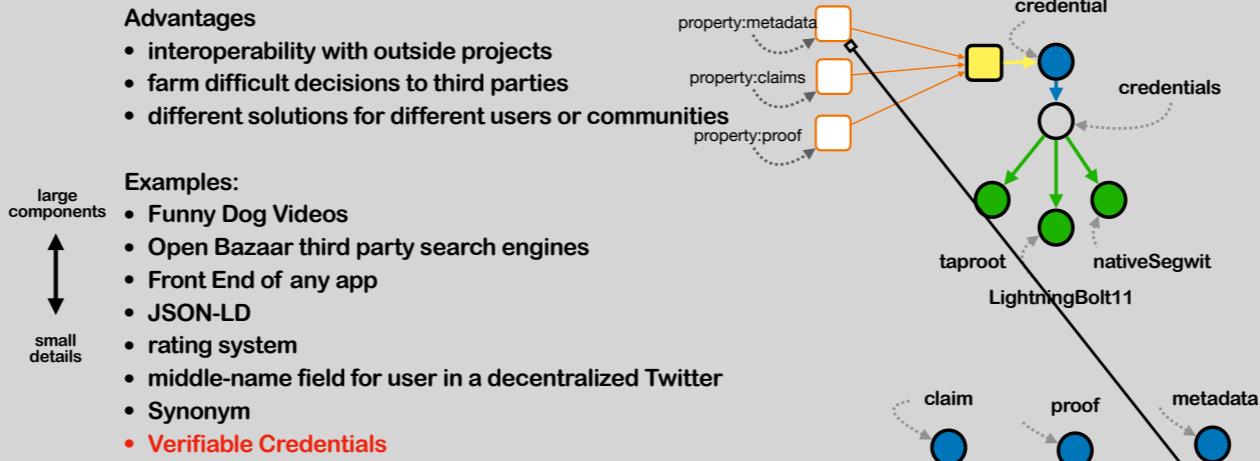
Examples:

- Funny Dog Videos
- Open Bazaar third party search engines
- Front End of any app
- JSON-LD
- rating system
- middle-name field for user in a decentralized Twitter
- Synonym

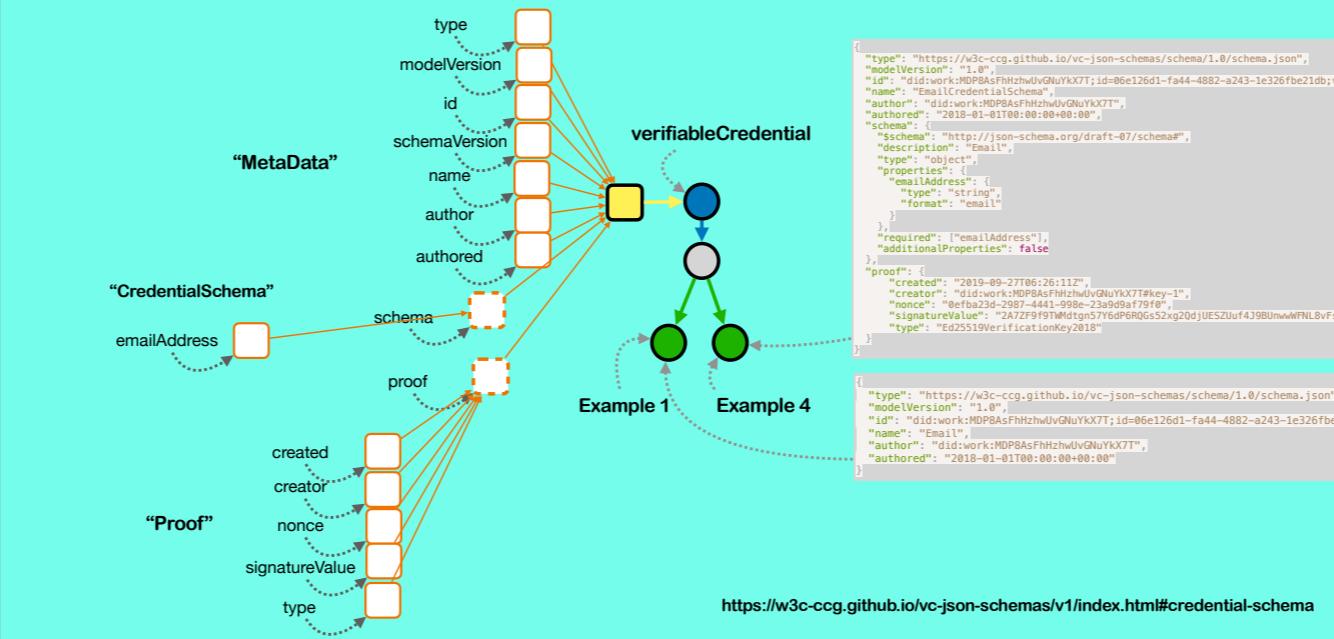
Should any part of my project be conceptualized?



Should any part of my project be conceptualized?

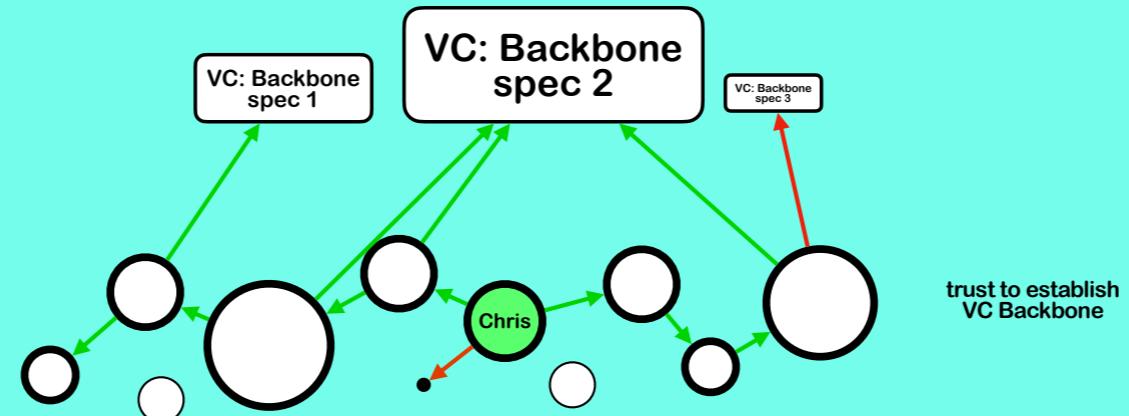


Should any part of my project be conceptualized? Verifiable Credentials



Verifiable Credentials

Influence Score:

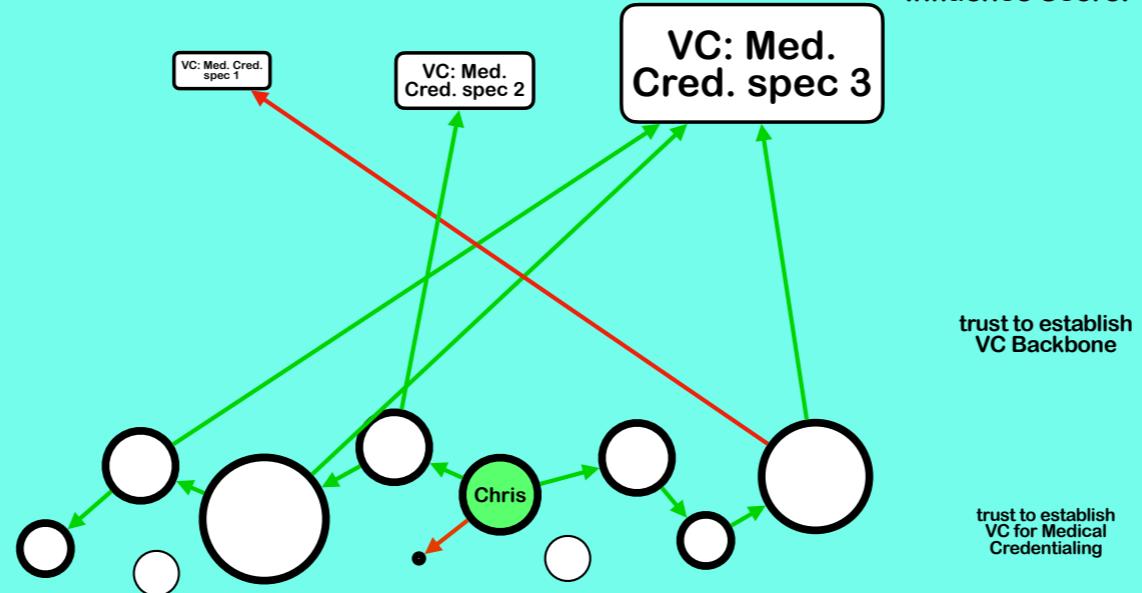


trust to establish
VC Backbone

trust to establish
VC for Medical
Credentialing

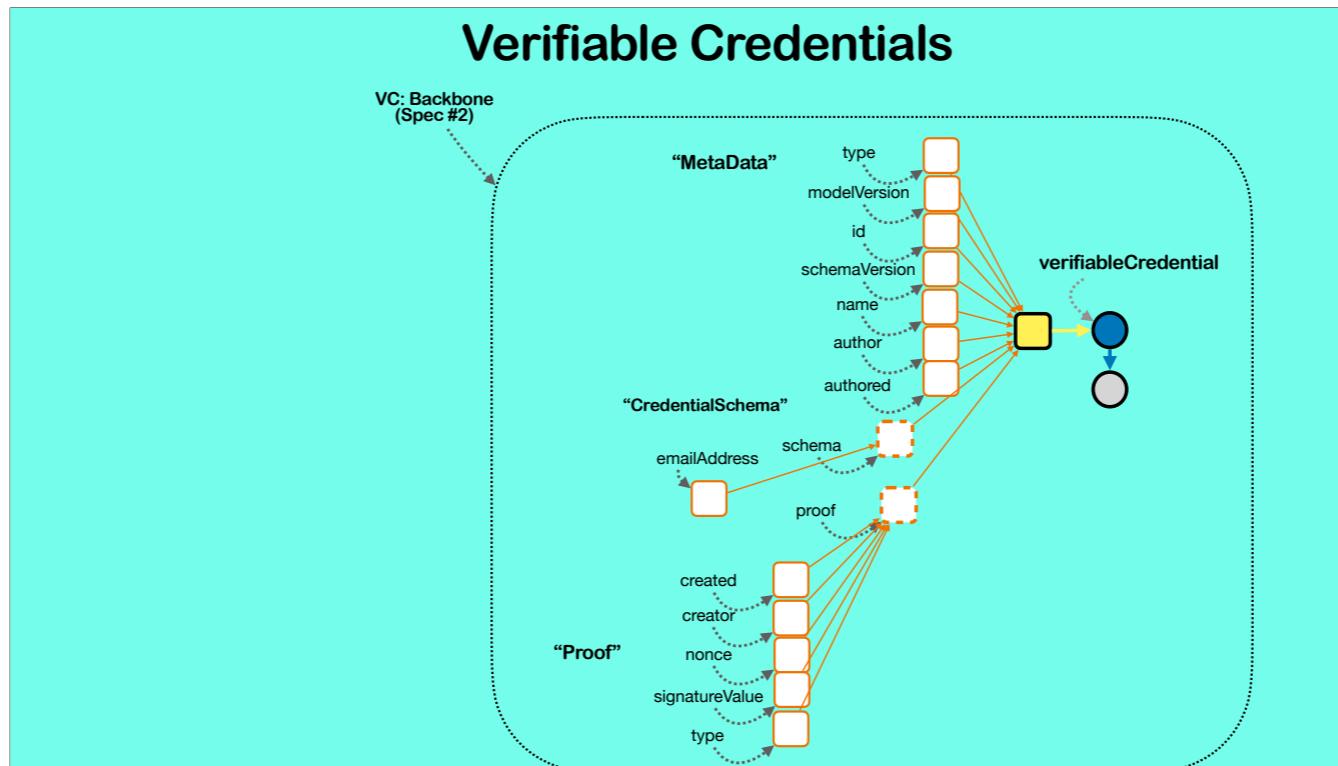
Verifiable Credentials: Layers of Influence

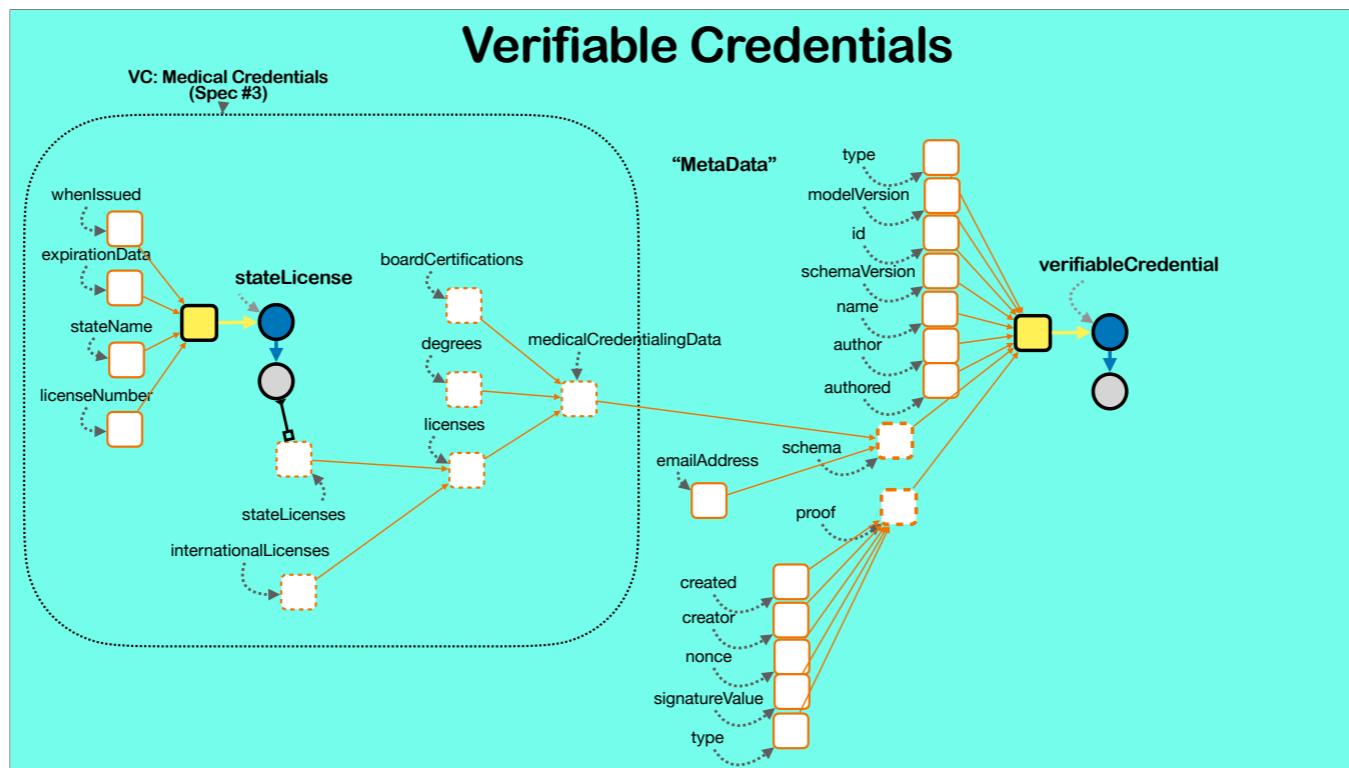
Influence Score:



trust to establish
VC Backbone

trust to establish
VC for Medical
Credentialing





UI for Verifiable Credentials

For the VC Committee member

- W3C committee members
- creates VC backbone + VC medical credential extension

For the active VC community

- vote on VC backbone and extension

For the users of VC

- chooses which VC to use
- physicians
- hospital administrators

UI for the VC Committee member

UI for the VC Committee member

the Grapevine

Hi Alice!

Profile

name

visibility

avatar

bio

keybase ID

IPFS ID

UI for the VC Committee member

 the Grapevine Hi Alice!

Communications Channels

Channel	Connect?	participants
IPFS		
• Public	✗	85
• VC Credentials	✓	11
GUN	✗	

Channel Connect? participants

IPFS		
• Public	✗	85
• VC Credentials	✓	11
GUN	✗	

UI for the VC Committee member

the Grapevine

Hi Alice!

Contacts List

	DIDs	IPFS	KeybaseID	normie IDs	normie Name
Alice				• gmail	
Bob		• IPFS		• Twitter	
Charlie		• KeybaseID			

UI for the VC Committee member

the Grapevine

Hi Alice!

Rating Templates List

- **Influence** (generic)
 - to evaluate Verifiable Credentials (backbone)
 - to evaluate Verifiable Credentials (medical)

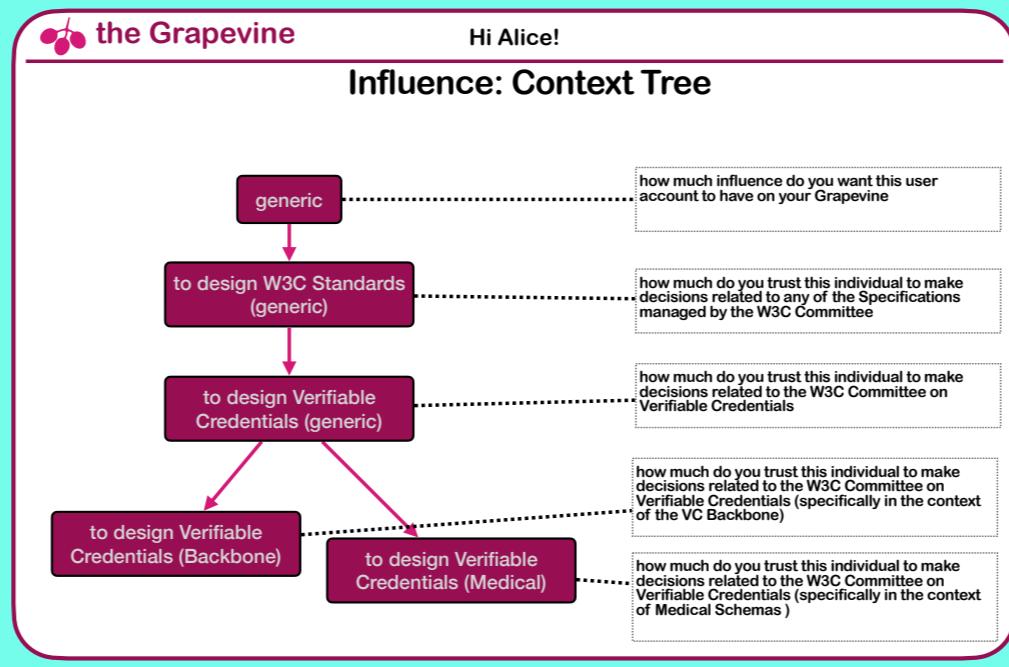
UI for the VC Committee member

the Grapevine Hi Alice!

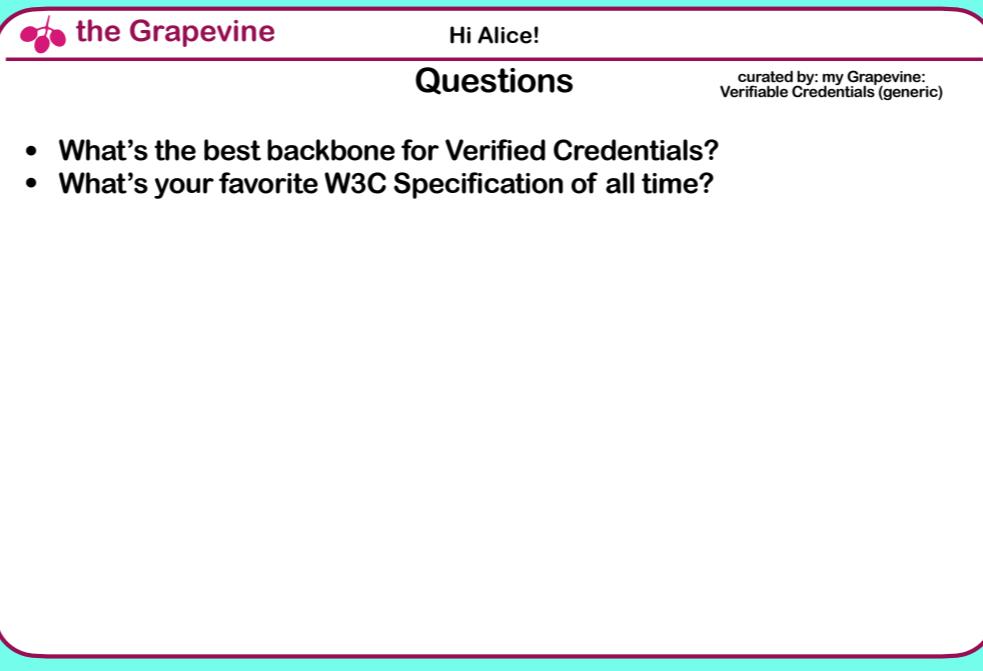
Composite Score Templates list

- Influence (generic)
to evaluate Verifiable Credentials (backbone)
to evaluate Verifiable Credentials (medical)

UI for the VC Committee member



UI for the VC Committee member



the Grapevine

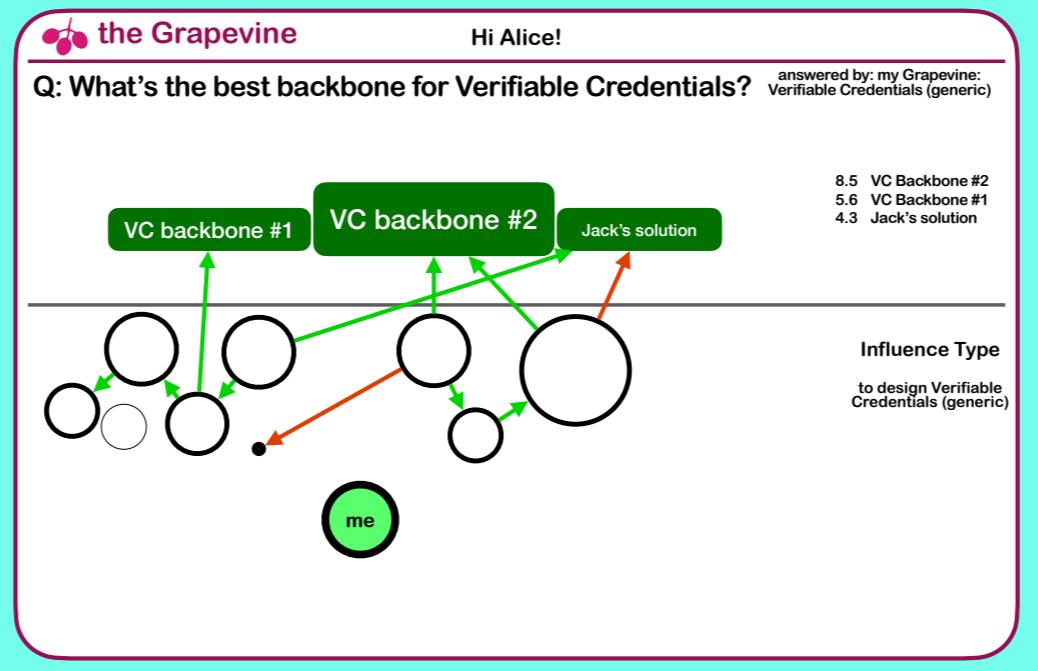
Hi Alice!

Questions

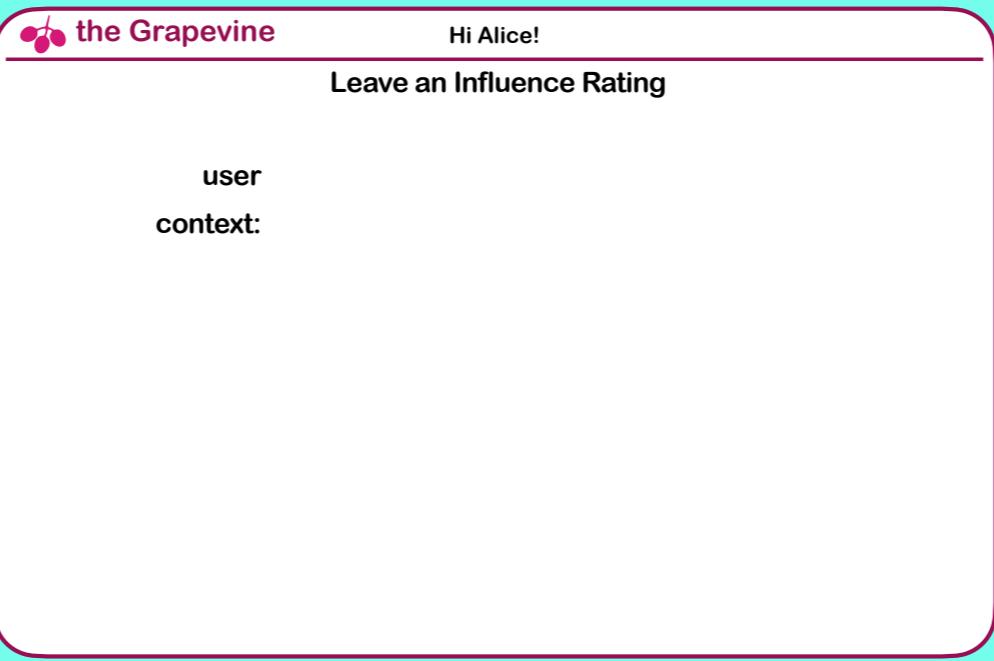
curated by: my Grapevine:
Verifiable Credentials (generic)

- What's the best backbone for Verified Credentials?
- What's your favorite W3C Specification of all time?

UI for the VC Committee member



UI for the VC Committee member



UI for the VC Committee member

the Concept Graph **Concept Graphs** Hi Alice!

- Verifiable Credentials (Backbone)
- Verifiable Credentials (Medical)

UI for the VC Committee member

Concept Graph:
Verifiable Credentials (Backbone) **Concepts List** Hi Alice!

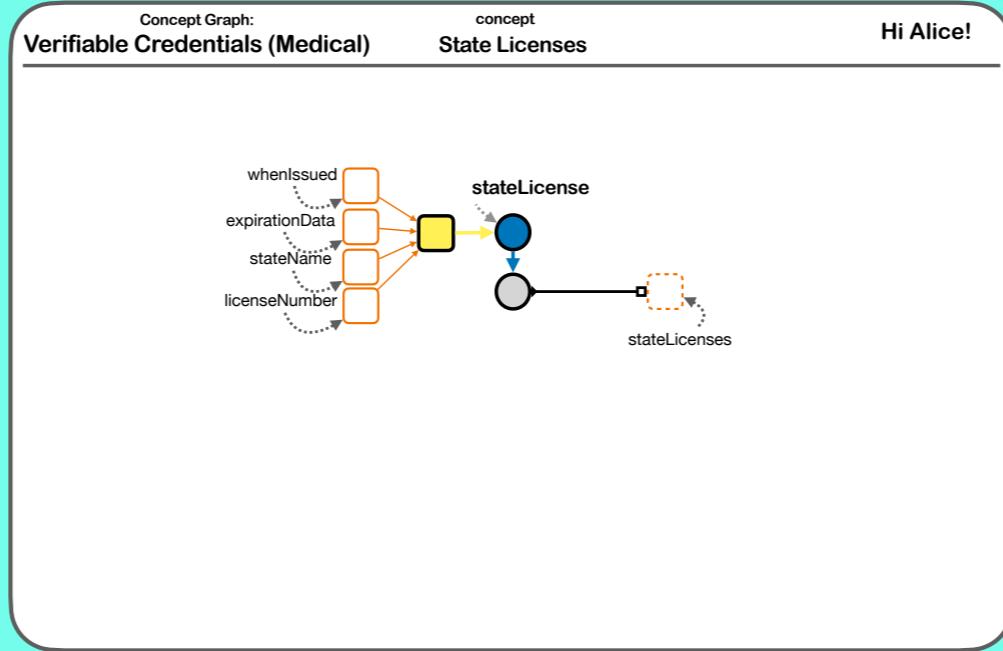
- Verifiable Credentials

UI for the VC Committee member

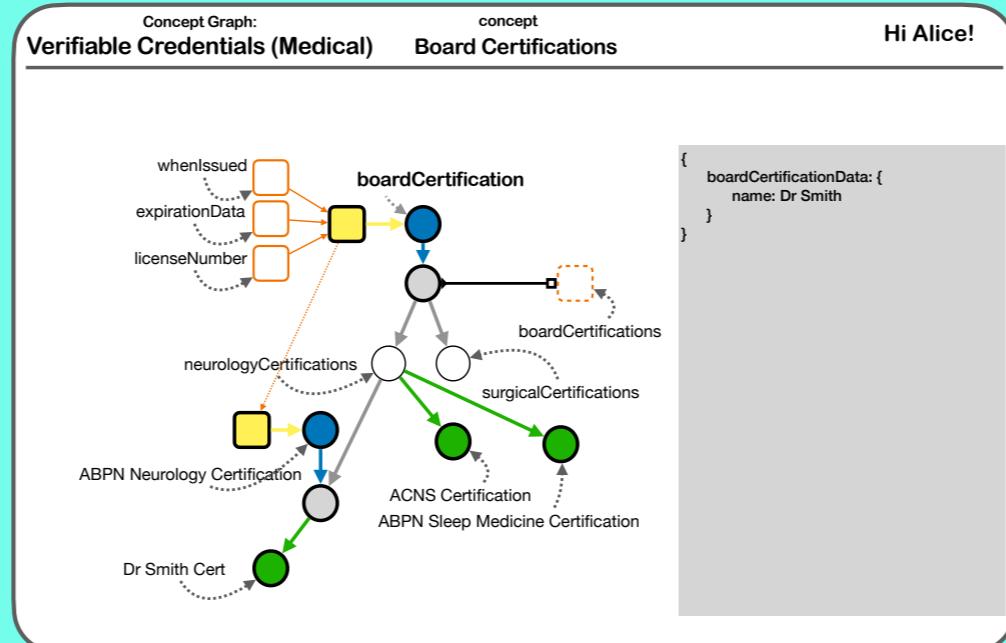
Concept Graph:
Verifiable Credentials (Medical) Concepts List Hi Alice!

- State Licenses
- Board Certifications
- Board Certification Types
- Professional Bodies
- Medical Schools

UI for the VC Committee member



UI for the VC Committee member



UI for the VC Committee member

Concept Graph:
Verifiable Credentials (Medical) concept
Board Certifications

Hi Alice!

Properties

- whenIssued
- expirationData
- licenseNumber

UI for the VC Committee member

Concept Graph:
Verifiable Credentials (Medical) Board Certification Types

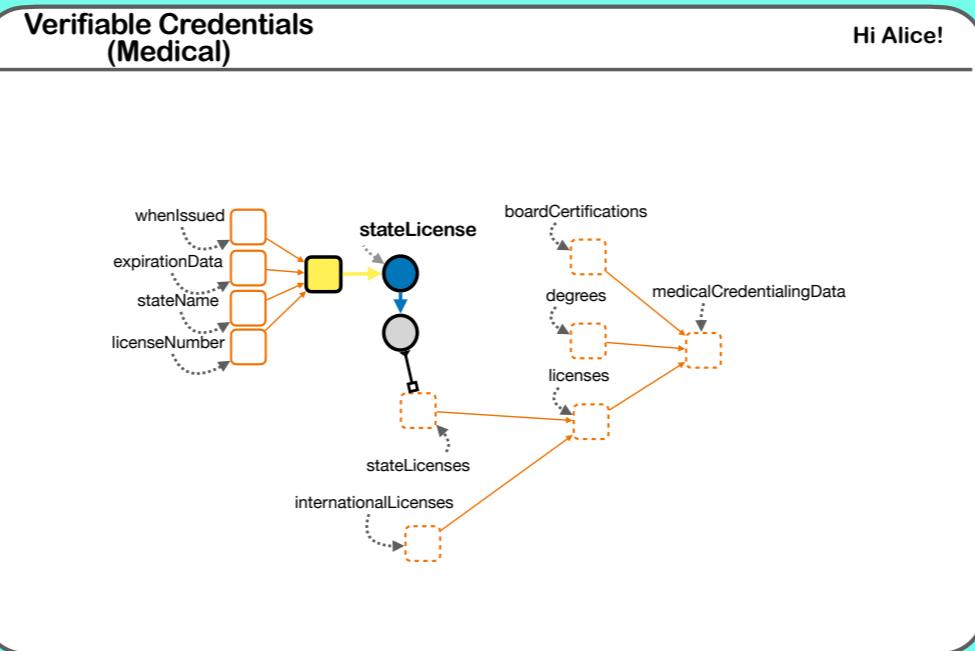
concept

Hi Alice!

The concept graph displays nodes and edges representing board certification types and their governing bodies. Nodes include 'boardCertificationType' (yellow square), 'ABPN Neurology Certification' (red-bordered yellow square), 'ACNS Certification' (green circle), and 'ABPN Sleep Medicine Certification' (green circle). Edges show relationships between 'name', 'governingBody', and 'url'. A tooltip for 'ABPN Neurology Certification' provides its JSON representation:

```
{  
  "boardCertificationTypeData": {  
    "name": "ABPN Neurology Certification",  
    "governingBody": "ABPN",  
    "url": "ABPN.com"  
  }  
}
```

UI for the VC Committee member

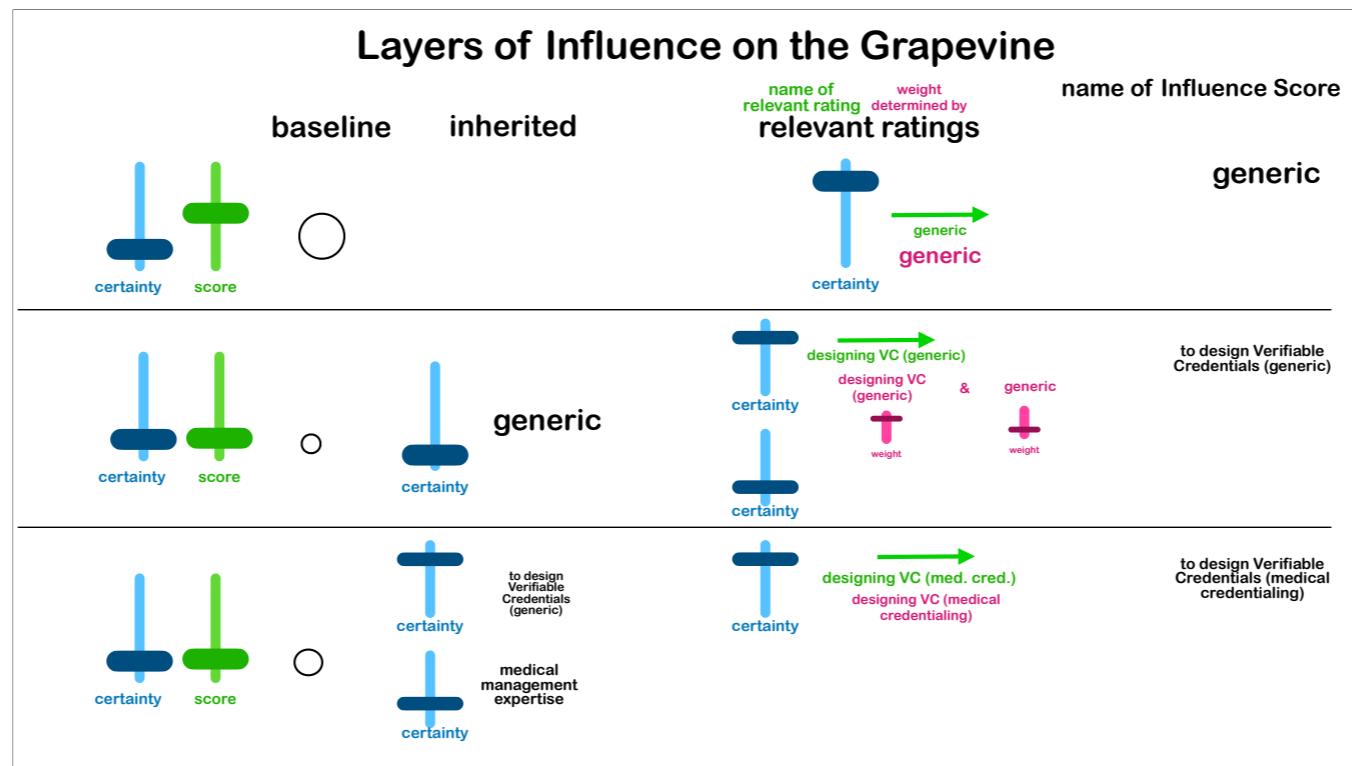


UI for the VC Committee member

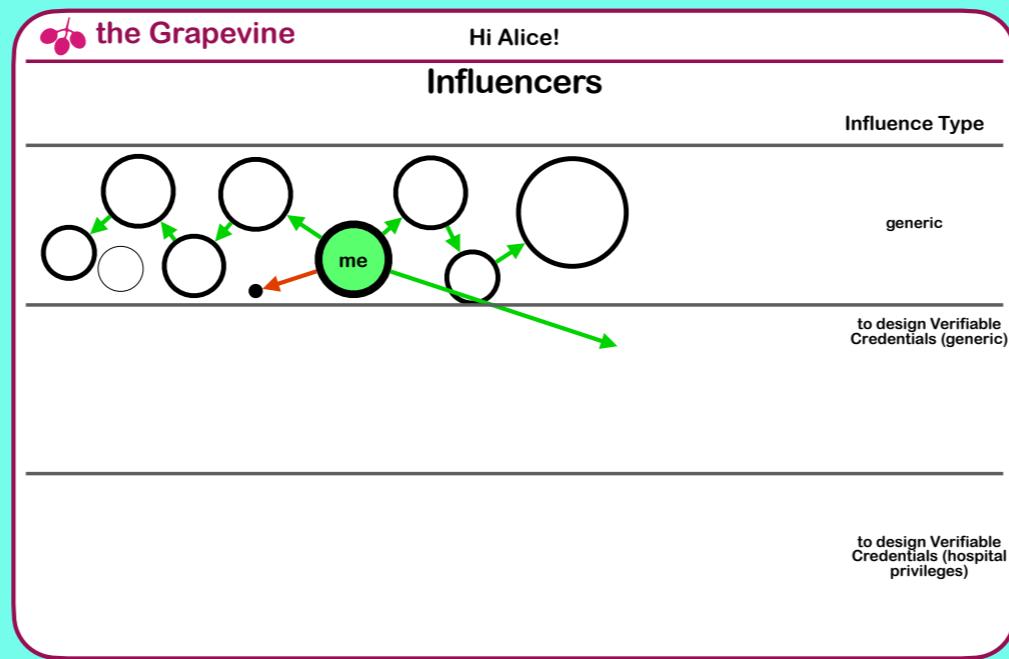
the Grapevine Hi Alice!

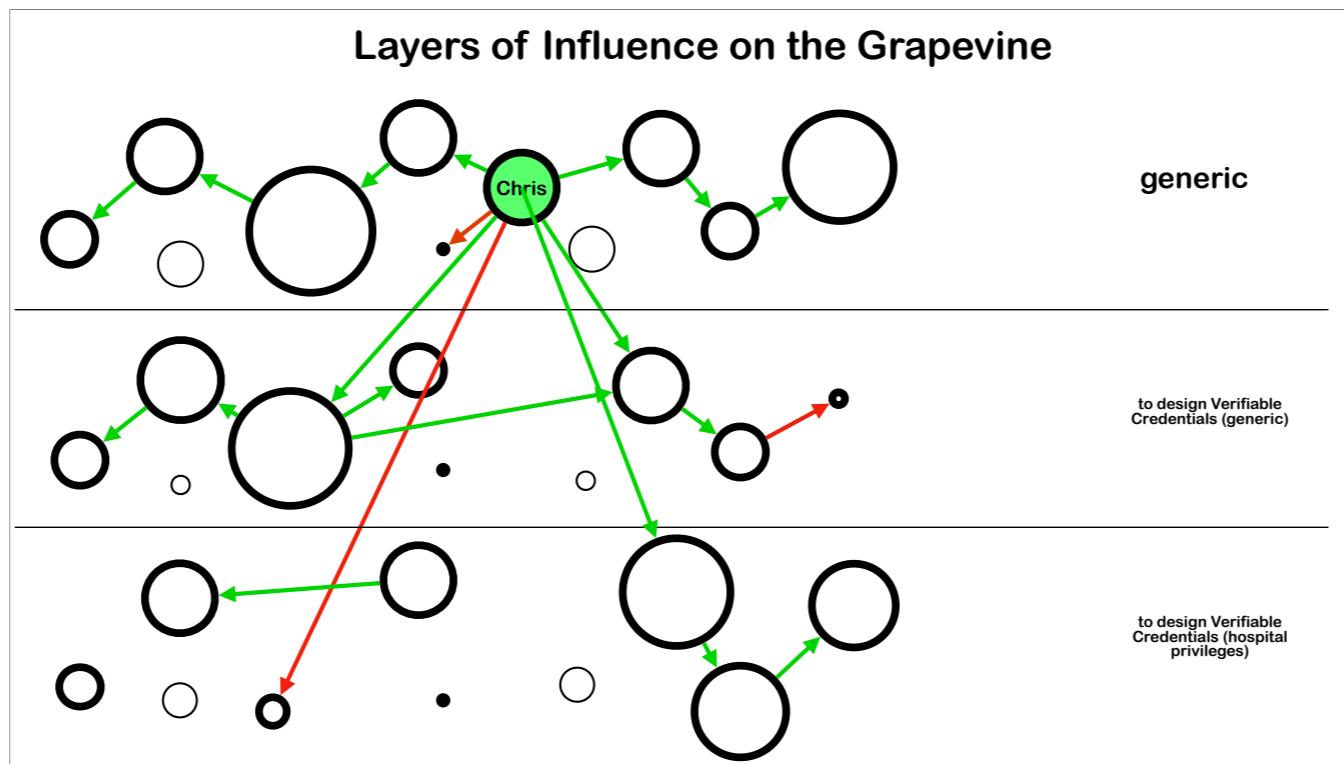
Influence Score Calculation [make new](#)

baseline	inherited	relevant ratings	Context
 weight	 score	<input type="radio"/>	set to default generic
 weight	 score	<input type="radio"/>	set to default to design Verifiable Credentials (generic)
 weight	 score	<input type="radio"/>	set to default to design Verifiable Credentials (hospital privileges)



UI for the VC Committee member

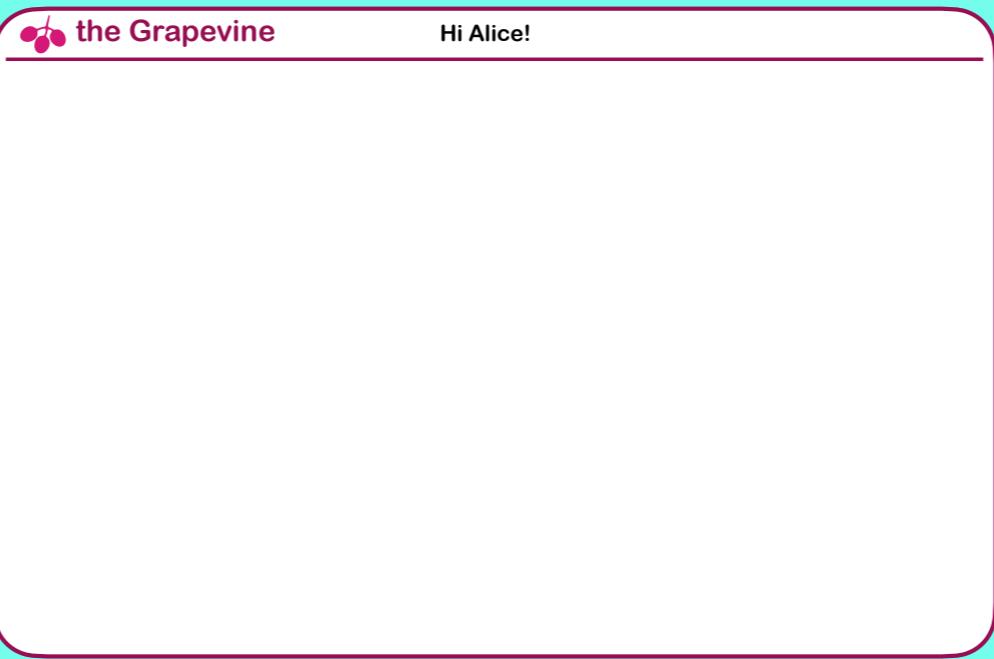




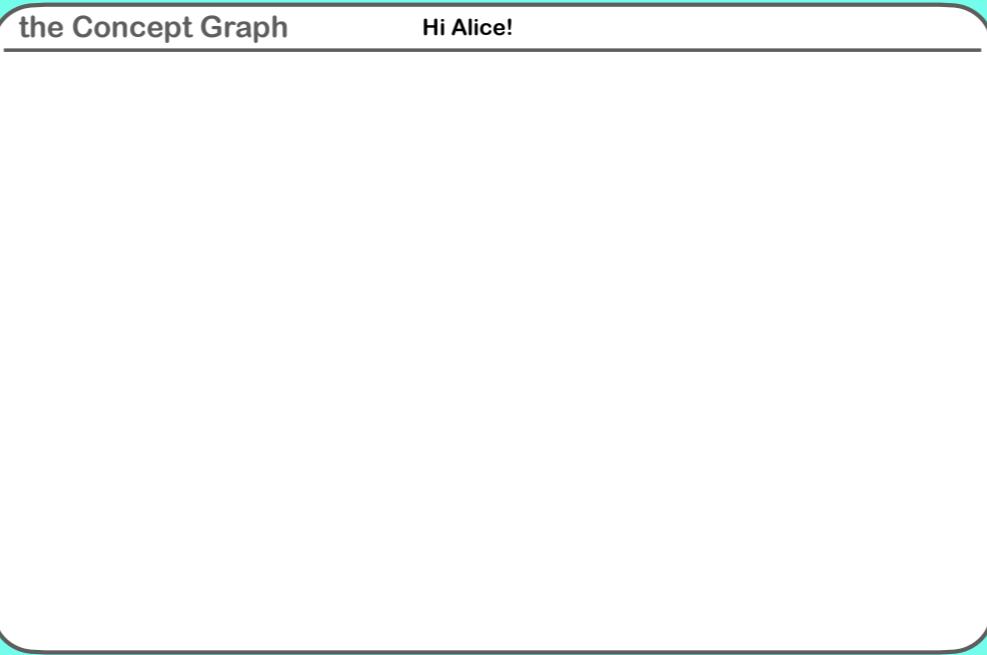
UI for the active VC community

UI for the Users of VC

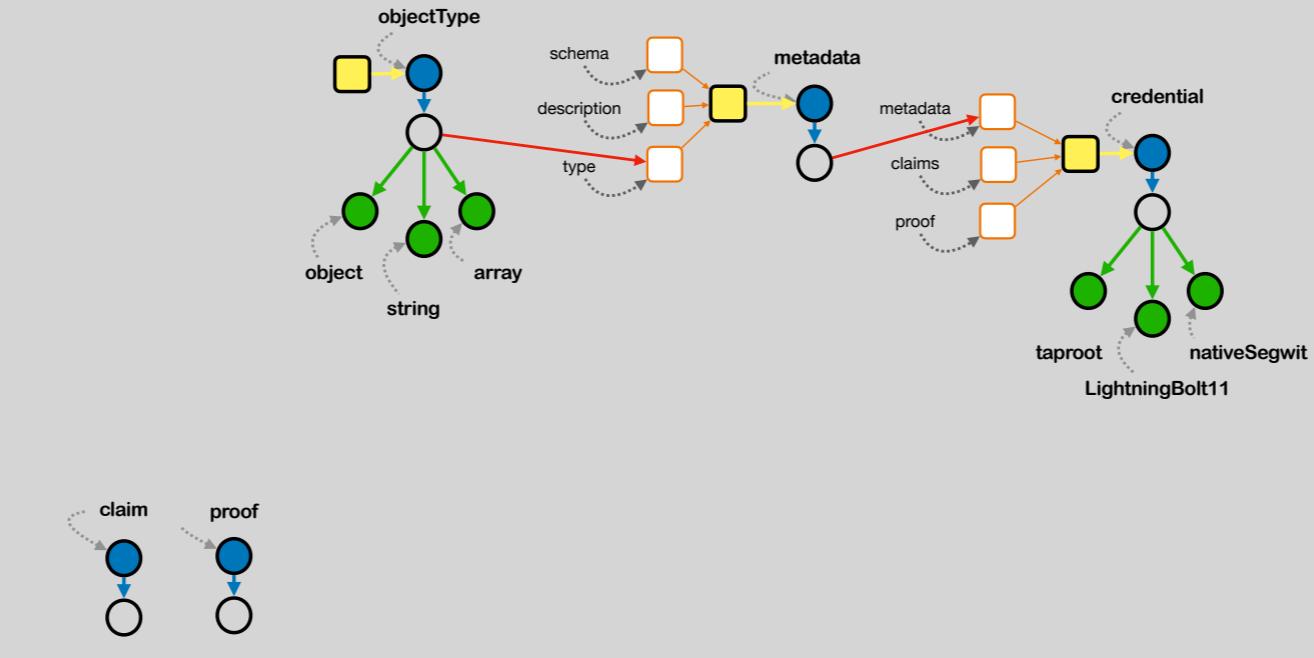
Top Panel of the Grapevine



Top Panel of the Concept Graph



Should any part of my project be conceptualized?



Should any part of my project be conceptualized?

Advantages

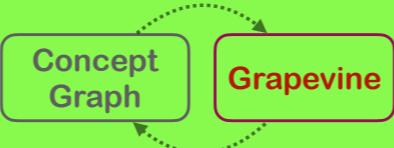
- interoperability with outside projects
- farm difficult decisions to third parties
- different solutions for different users or communities

So to summarize: think about whether your app could be conceptualized. And what advantages it might confer. Especially if you want to have a rating system.

**For much of the Grapevine and Concept Graph,
progress can proceed incrementally.
Utility exists even at early stages.**

- Concepts
- Context trees
- Reputation systems
- Conceptualization

INTEROPERABILITY is the key



Regarding the two apps: In this talk I will be describing two apps that I have been building, called the CG and the Grapevine. These, paired together, represent my best attempts on how to implement the Loki Principle towards the dWeb. the point of my presentation is not to tell you that you need to adopt my two apps - although that would be cool! Rather, I am trying to sell you on the Loki Principle. I feel kinda like Jack Mallers building Strike from things I've heard him say on podcasts, things I agree with: I'd love it if you use the apps that I build. That's my goal. But ultimately, best of all possible worlds, suppose I build a team, build tools that gain wide acceptance: wonderful! But I fully expect other people to come up with competition for the CG and Grapevine. Competing implementations of the Loki Principle. Maybe very similar to the CG/Grapevine but better design; or maybe someone will think of ways to implement it that I haven't. So as I describe the CG and the Grapevine, you may find yourself thinking, maybe you don't agree with some of my design decisions, or the choice of tech stack that I'm using; (I am partial to the IPFS for data storage, bitcoin, lightning network for payments, json for data files, JSON Schema, graph databases, neo4j as an open source graph database tech stack); maybe you choose different tools; that's OK. The reason I'm explaining the CG and Grapevine in detail is because that is the best way for me to explain the Principle. I have an abstract idea, the LP, and here are the concrete specifics of how I implement this idea, the CG/G. Maybe you can come up with a better implementation. Good. I want competing implementations. The world needs it. And ultimately, they're all going to have to be interoperable. Just like competing implementations of the LN are interoperable. If they're not interoperable, then that defeats the whole point; we've failed. bc that's the whole point of what I'm doing: interoperability. This is part of why I'm not introducing a blockchain. I want interoperability; I want you to buy in to the abstract idea of the LP, and i want interoperability, but I don't want it to require you to have to commit to any one particular tech stack.

The Cerebral Cortex and the Decentralized Web

**a Thousand Brains ... a Thousand Platforms?
Shared Challenges, Shared Solutions
(and Social Constructs!)**

David Strayhorn MD, PhD

When I say consensus, I'm not talking about a solution to the Byzantine General's problem or an alternative to Nakamoto consensus; it's a completely different kind of consensus, something I am calling Loose consensus. Loose consensus is related to the idea of a social construct. So anyone who has taken one too many philosophy courses and has wondered what's the deal with social constructs, i'm going to give you a new way to think about that question.

The Decentralized Web and the Cerebral Cortex

a Thousand Brains ... a Thousand Platforms

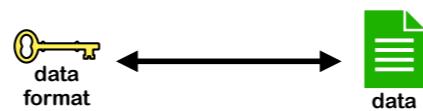
Shared Challenges, Shared Solutions

David Strayhorn MD, PhD

Hello everyone. In this video I am going to be describing a method of organizing information, one that I think ought to be incorporated into the decentralized web; and one that I speculate may already be incorporated into the cerebral cortex. As I will explain, I think the cortex can be modeled as a decentralized system; and, as decentralized systems, the cortex and the dWeb share a common problem: which is inability for entities to establish a shared language for communication with each other. The purpose of the method that I will propose will be to allow entities in a decentralized system to arrive at a minimum level of consensus, something I'll call Loose Consensus, on a common language so that they can communicate efficiently and effectively with each other.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.



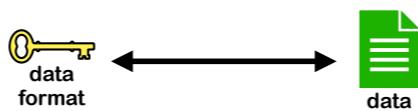
Implementation:



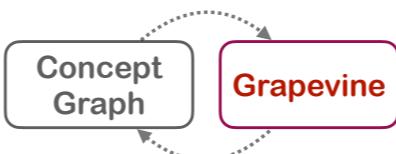
But it's possible to get these details wrong. The grand strategy needs to be properly envisioned first. And so they need to be developed carefully, conceived properly, in a well thought out manner, before embarking prematurely. Figuring this out early is important to prevent lost time and wasted efforts on the part of developers.

Solution: Principle of Loki

For any piece of data, there must always be a simple, straightforward method to connect that data to a record of its data format.

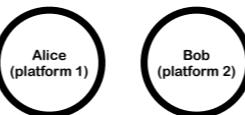


Implementation:



"Loose consensus" over a certain class of questions:

- mundane; not controversial
- no Schelling point
(e.g.: how to format data!)



"Socially Constructed"
Consensus

I've compared LC to a language. I might even go so far as to say that Loose consensus, effectively, is what some postmodern philosophers might call a social construct.

The Principle of INCREMENTAL LEARNING

A surprising number of design decisions are impacted by the need to accommodate INCREMENTAL updates to the topological database.

Coarse Graining versus Fine Graining must be allowed to change INCREMENTALLY without breaking things

Examples:

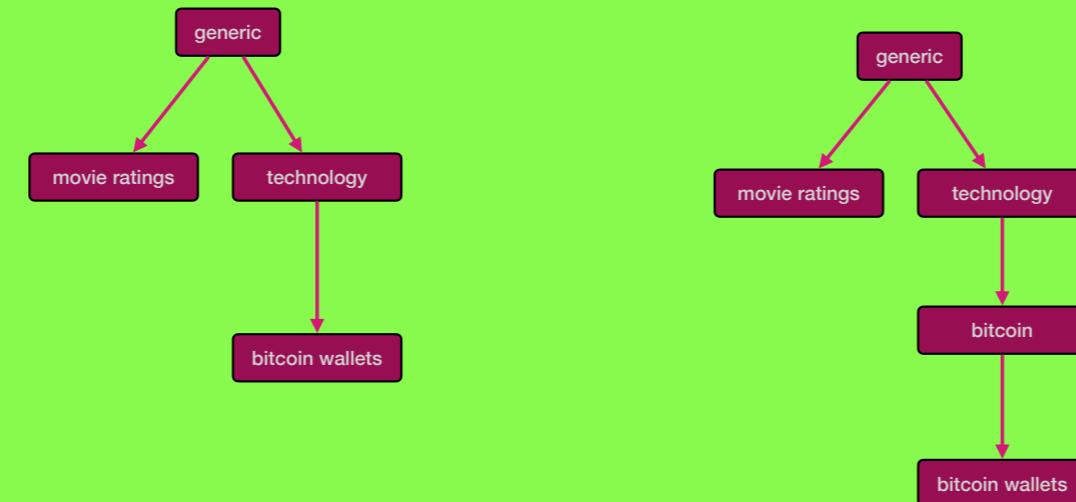
- **Concept Graph:** dogData and animalData: should the dogData property be sub-property of animalData? answer: usually no! bc think about what happens if add/subtract subset concepts?
- **Context tree** must be allowed to be updated INCREMANTALLY. This constrains the design of default and inherited Grapevine Influence scores

INEVITABILITY

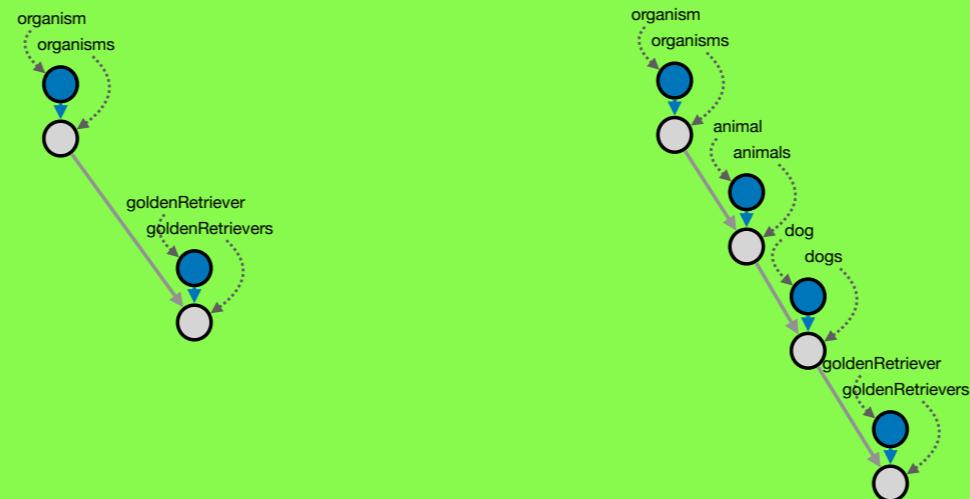
Design decisions are INEVITABLE. But first we must adopt the following principles (and take them seriously):

- Cypherpunk concept of privacy (pseudonymity, portability, etc)
- Principle of Loki; JSON files; graph database; Loki Pathway
(JSON Schema & Property Tree follow naturally from the above?)
- Principle of INCREMENTAL LEARNING
- Superpositioning of graphs

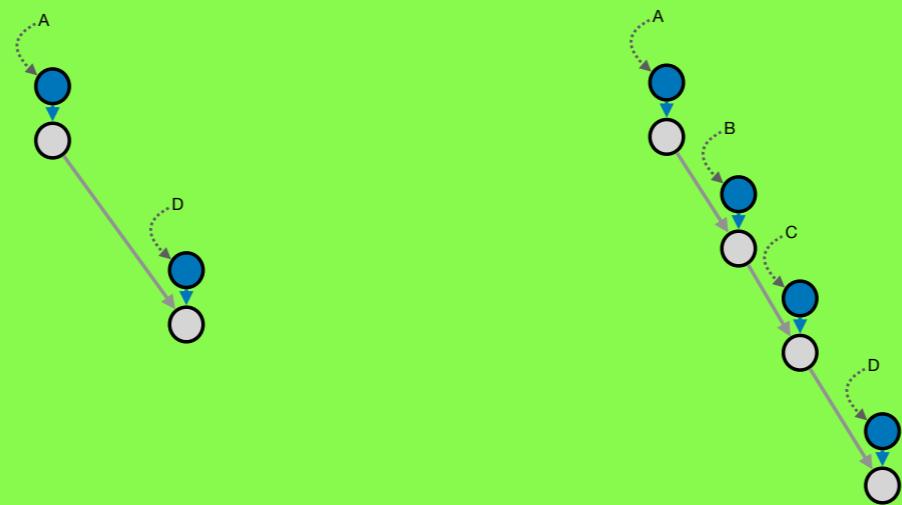
INCREMENTALISM



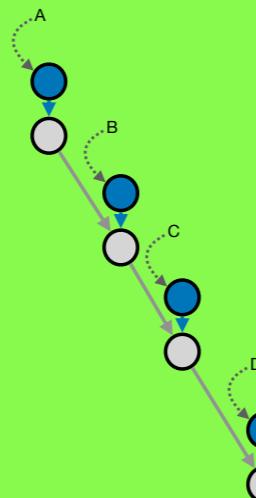
INCREMENTALISM



INCREMENTALISM



INCREMENTALISM



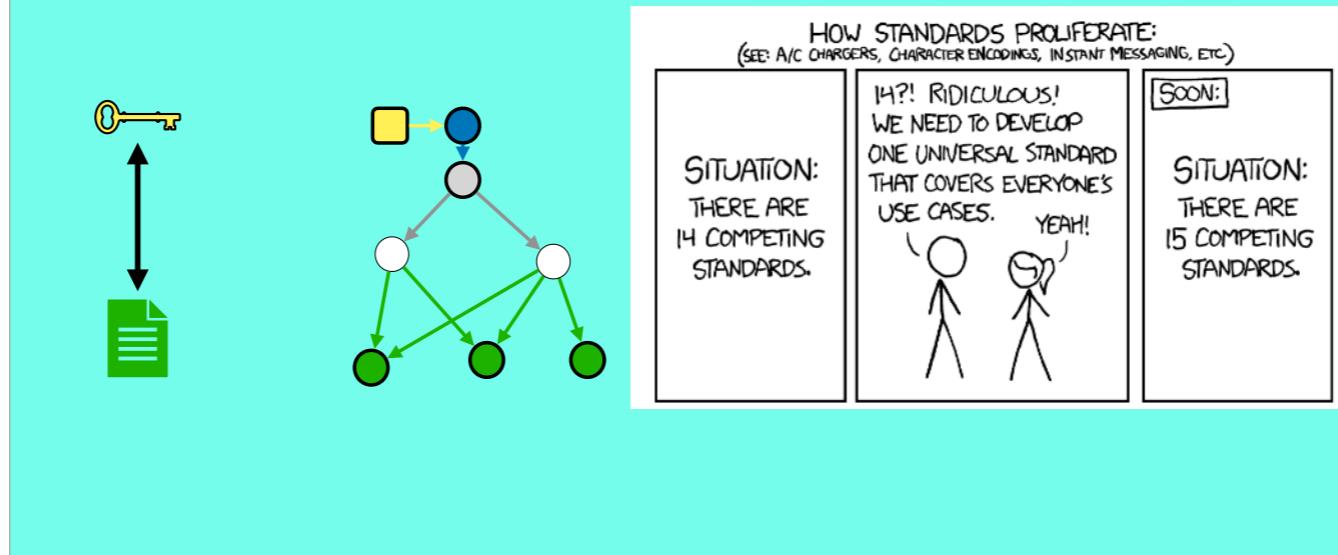
Proposed Solution #1: Just make sure to do
EVERY level CORRECTLY the FIRST time.

Problem: THERE IS NO SUCH THING.

This solution is unacceptable, unless you wish to
abandon the principle of INCREMENTAL
LEARNING.

Imagine dWeb without the PoL / Loose Consensus

Principle of Loki: For any piece of data, there must always be a simple, straightforward, well defined method to connect that data to a record of its format.

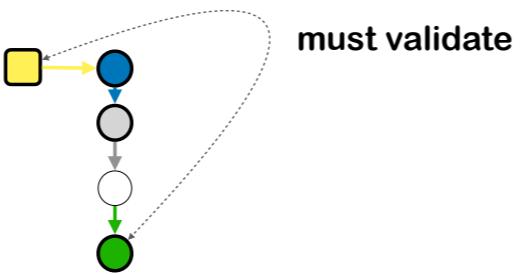


To anybody who is working on any project related to the dWeb. I challenge you to ask yourself: do you think it is possible for this world to exist without some version of the PoL having been put into place? Its absence would mean that for any given piece of data, a record of its expected format is where exactly? Maybe you envision it to exist in the perpetually poorly maintained documentation of open source repos or project websites? Maybe it's in the minds of god and a few developers? Maybe we just need to make one standard that we can all use. That sure would be awesome, wouldn't it? If that's what you think, then what is your answer to XKCD?

(duplicate) Envision a world where the dWeb is a reality. It has lived up to its promise. It has gained so much traction, that Big Tech platforms have faded into obscurity. Now I challenge you to ask yourself this question: Where do you find formatting information? Where is it recorded? For any given piece of data, is it scattered among open source repos, poorly documented, in the heads of god and a few developers?

Internal Data Inconsistencies

Loki Consensus



Any failure to validate = breaking of Loki Consensus

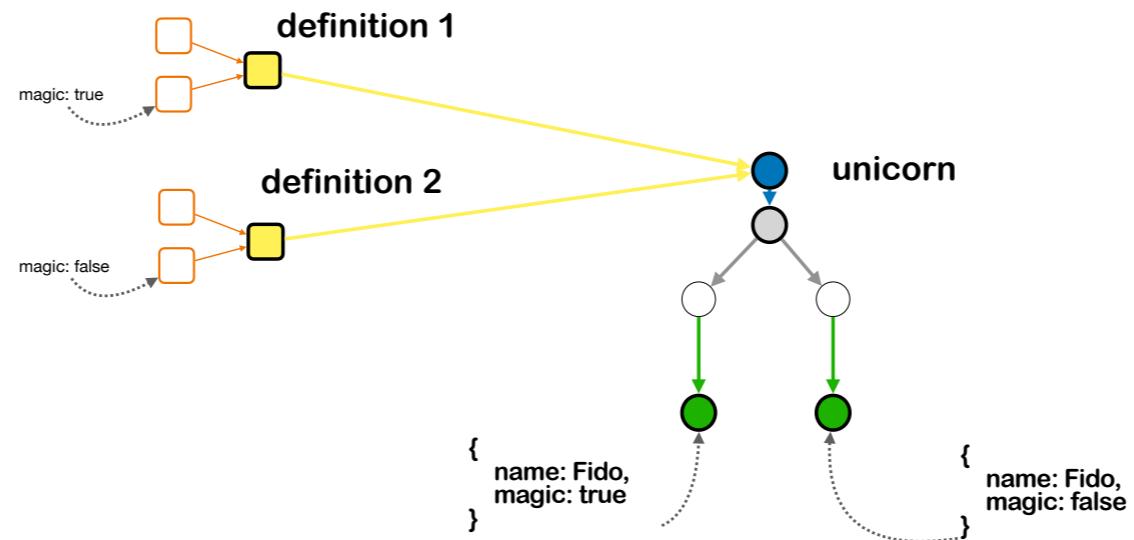
Loki Scar: entrenched instance of Loki antiValidation

EQUIVOCATION

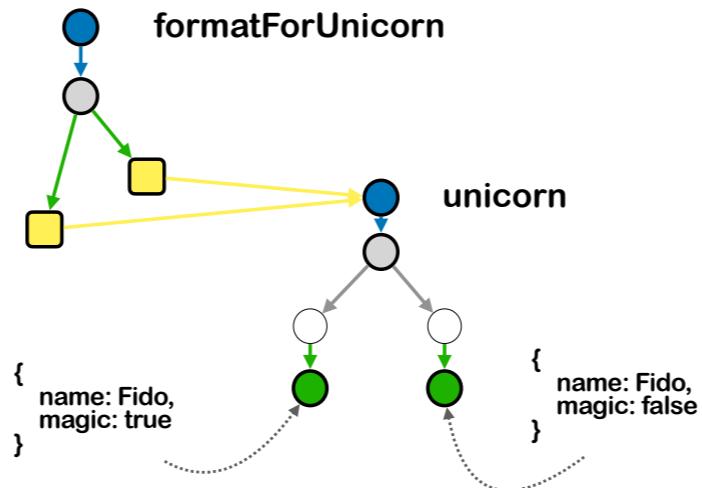
EQUIVOCATION == changing from one definition to another

dishonest equivocation: using equivocation to establish Loki Scar

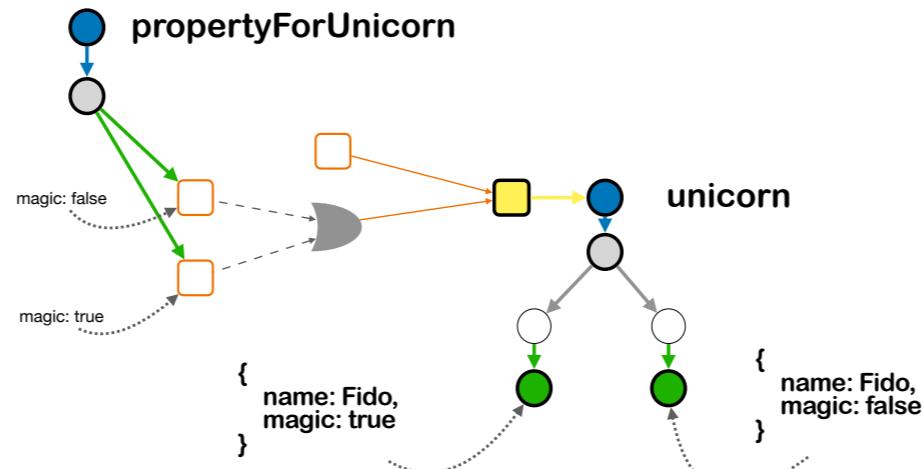
Self-Deception via Equivocation

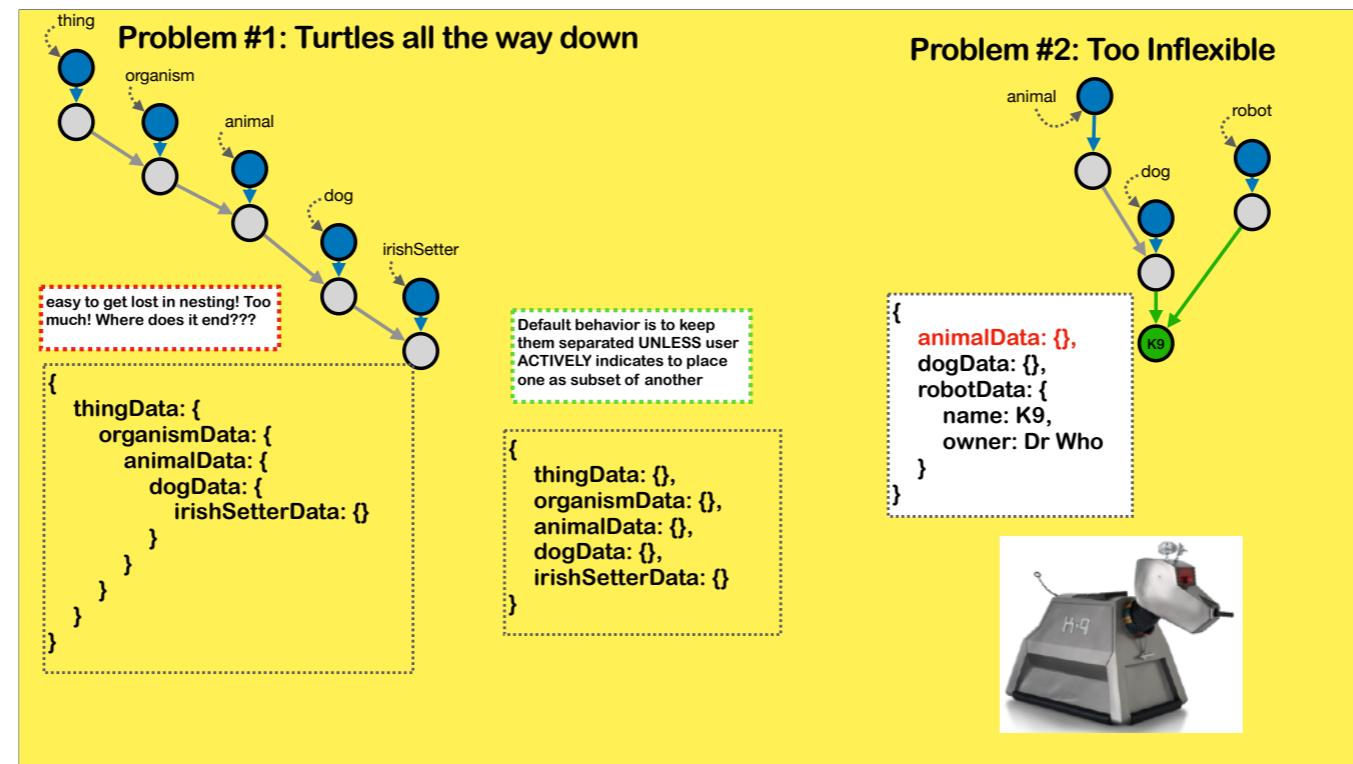


(Self-Deception via) Equivocation

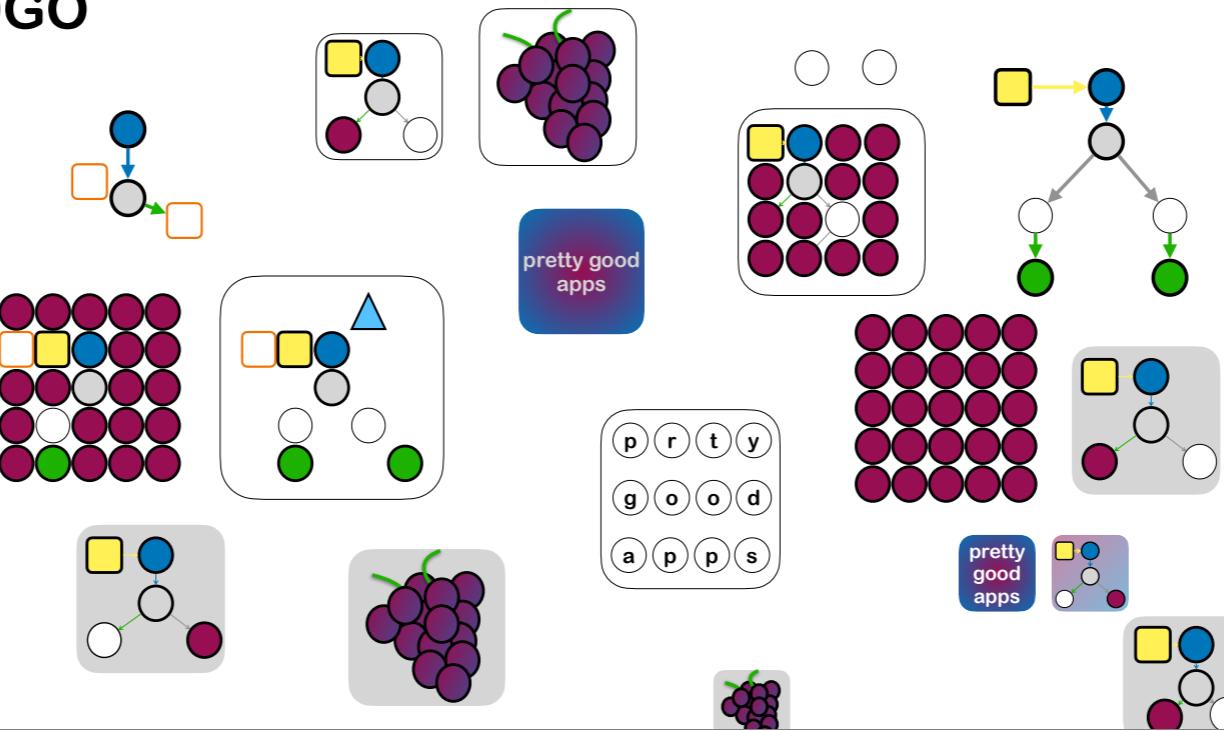


(Self-Deception) via Equivocation

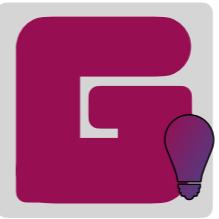
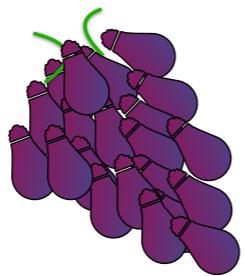
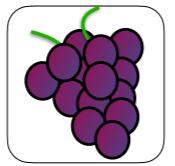


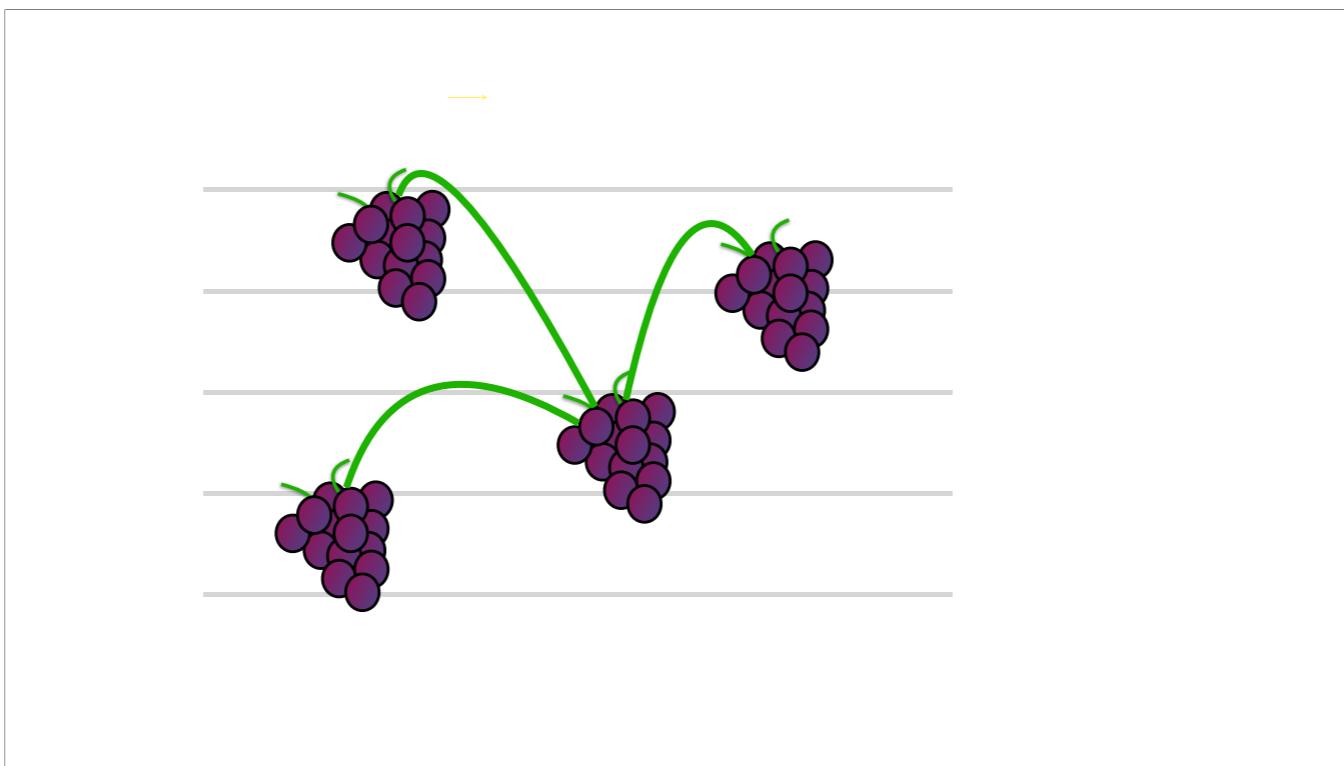


LOGO



LOGO





PLEX

LOGO

