Define Syntax-Locked and Syntax-Free Artifacts: Clearly articulate what constitutes a syntax-locked artifact versus a syntax-free artifact, providing comprehensive definitions and characteristics for each.
Compare and Contrast Artifacts: Systematically compare syntax-locked and syntax-free artifacts across various dimensions, such as complexity, flexibility, performance, ease of use, and applicability in different contexts.
Develop Evaluation Criteria: Establish rigorous, objective criteria for evaluating and measuring the effectiveness, efficiency, and suitability of syntax-locked and syntax-free artifacts in various scenarios.
Empirical Analysis: Conduct empirical research to gather data and insights on the real-world application and performance of both types of artifacts. This involves designing experiments, collecting data, and performing statistical analyses to validate findings.
Impact Assessment: Assess the impact of syntax-locked and syntax-free artifacts on different stakeholders, including developers, end-users, and organizations. Consider factors like learning curve, adaptability, and long-term sustainability.
Provide Guidelines and Recommendations: Based on the findings, offer practical guidelines and recommendations for choosing between syntax-locked and syntax-free artifacts. Address potential trade-offs and suggest best practices for different use cases.
Identify Future Research Directions: Highlight gaps in the current understanding and propose areas for future research to further explore the nuances and implications of syntax-locked and syntax-free artifacts.

```
// Airtable entity
Table Airtable {
  AirtableId int [pk, unique] // Primary Key
  Ideas Idea[] // Relation to Ideas table
  IdeaTransformers IdeaTransformer[] // Relation to IdeaTransformers table
  Generations Generation[] // Relation to Generations table
  GenerationTransformers GenerationTransformer[] // Relation to GenerationTransformers table
  Entities Entity[] // Relation to Entities table

  Note: "Airtable entity containing metadata and relationships to ideas and transformations."
}

// Ideas entity
Table Idea {
  IdeaId int [pk, unique] // Primary Key
  SourceIdea string // Source idea text
  Name string // Idea name
  IsActiveIdea string // Status of the idea
  Generations string[] // List of generations associated with the idea
  GenerationTransformers string[] // List of generation transformers associated with the idea
  GenerationTransformerFullPrompts string[] // Full prompts used for generation transformers
  GenerationTransformerNames string[] // Names of generation transformers
  AirtableId int [ref: > Airtable.AirtableId] // Foreign Key to Airtable

  Note: "Idea entity representing individual ideas, their statuses, and relationships."
}

// IdeaTransformers entity
Table IdeaTransformer {
  IdeaTransformerId int [pk, unique] // Primary Key
  Idea string // Associated idea
  IsActiveIdea string // Status of the idea
  IdeasSourceIdea string // Source idea text
  IdeaName string // Name of the idea
  AirtableId int [ref: > Airtable.AirtableId] // Foreign Key to Airtable
  TransformedArtifacts string // Transformed artifacts details
  FullPrompt string // Full prompt used for transformation
  Name string // Transformer name
  fields string // Additional fields

  Note: "Transformer entity for ideas, storing transformation details and prompts."
}

// Generations entity
Table Generation {
  GenerationId int [pk, unique] // Primary Key
  GeneratioNumber int // Generation number
  Idea string // Associated idea
  Name string // Generation name
  IsActiveIdea string // Status of the idea
  IdeaName string // Name of the idea
  IdeaSourceIdea string // Source idea text
  AirtableId int [ref: > Airtable.AirtableId] // Foreign Key to Airtable
  TransformedArtifacts string[] // List of transformed artifacts

  Note: "Entity representing generations of ideas, tracking status and transformations."
}

// GenerationTransformers entity
Table GenerationTransformer {
  GenerationTransformerId int [pk, unique] // Primary Key
  Transformer string // Transformer details
  Generation string // Associated generation
  TransformedArtifacts string[] // List of transformed artifacts
  GenerationIdea string // Idea associated with the generation
  GenerationName string // Name of the generation
  Name string // Transformer name
  IsActiveIdea string // Status of the idea
  GeneratioNumber int // Generation number
  GenerationIdeaName string // Name of the idea associated with the generation
  GenerationSourceIdea string // Source idea text
  TransformerFullPrompt string // Full prompt used for transformation
  Prompt string // Transformer prompt
  AirtableId int [ref: > Airtable.AirtableId] // Foreign Key to Airtable

  Note: "Entity representing transformers applied to generations, tracking transformation details and prompts."
}
```

```
// Entities entity
Table Entity {
   EntityId int [pk, unique] // Primary Key
   Name string // Entity name
   PluralName string // Plural form of entity name
   AirtableName string // Associated Airtable name
   AirtableId int [ref: > Airtable.AirtableId] // Foreign Key to Airtable

   Note: "General entity storing metadata for various entities within Airtable."
}
```

This GPT is to help me write this paper.

Research Paper

So - I want to write a paper. TEll me about the notion of syntax-locked vs syntax-free models. I've been talking to my Professor of MDE and the most recent language that he has started using is:
Original Position: Everything is is a language, one person's WHAT is another person's HOW, and so there is no bright red line between them, it is pink, and blurry, at best.

1) I accept that an ACID compliant DB is not "a language" (i.e. not EVERYTHING) is a language
 - there is the caveat that while he accepts that they are different, he insists that this difference is a "red herring"
2) Syntax-locked (i.e. 1d text files that require parsing, lexting, tokenizing, interpretation,etc before they can be understood) is a concept that _I think_ he is basically on board with.
3) The idea that there is a bright red line is still pink because he said on our last call, repeatedly, that he still doesn't really know/understand what I mean when I say syntax free.

So - tell me about the difference(s) between syntax-locked vs syntax-free content.
So - this is actually a statement about an aspect/element of language that I feel like is largely A) not yet understood, B) as a result, mostly unexplored scientifically.

So - here are the buckets that I would like to isolate, scientifically.

I would like the paper to be based on potentially one, but alternatively, a set of specific predefined prompts - for this discussion let's assume there is always 1 ROOT prompt that then has a cascading series of analytics performed on it. An example of such a prompt is:

Need: To do list app, with a few features, with some examples of categories and to do items.

GPT 4 would be the "objective" test tool. It's obviously biased itself, but I think that for the kinds of changes that we want to study, these biases should not play a part.

I want to develop a series of automated test protocols that can, perform tests like this:

1) ask gpt4 to answer this prompt 1000x:

Write a 1 paragraph description of all requirement details, features, categories/to do items. Be as complete and detailed as possible.

2) ask gpt4 to answer this prompt 1000x:
Write a single-source-of-truth.json with a list of all requirement details, features, categories/to do items. Be as complete and detailed as possible.

This would be the raw data that we would further analyze, also using basic data science along with continuing to leverage GPT 4.

Here are some examples of the kinds of metrics that I would like to measure.

1) We would ask GPT (per natural language/json responses - maybe multiple times for each) to generate a list of the top, say, 10 keywords (names, due dates, statuses, etc) in the content. This would be added to the core source data

2) Further analysis could, for examples, start analyzing:
 - what are the total number of synonyms (matched ideas) across the entirey study
 - by nlp/vs
 - by any other relevant slices/dices through the data

3) Then - we would extend all of the data by 1 generation, and repeat the process. In other word, we would start a new conversation with the 1st generation response as the input for the 2nd generation, with a prompt asking it to describe that either in natural language english, or as a single source of truth. So each previous output node would fork, and we would now have 6K data sets, each with a hierarchy. Except now, we can add an entirely new layer of analysis. Specifically
 - not just what is the total size of the vocabulary used in various sub-sets of the data, but also a differential between the 2nd generation and the 1st.
 - We can also track (statistically) how likely it is that a key term/idea/concept will be preserved (to do due-date vs to do deadline - same idea - totally different word). How likely is it that new ideas will be inserted, or dropped version to version.
 - Each node could either fork to nlp/json like originally done, but it could also include 1 or more languages that are designed to capture those rules.
 - Abstract questions like:
    "How many features are described?"
    "Does the app support app completion?"
    "Does the app track when tasks are due?"
    "Are there deadlines?"

4) this process could be repeated for N generations - and the metrics would be tracked and visualized generation to generation. It would generate SOOOOOOO much data, right?

5) At each generation, changes could be introduced like "Add the ability to track how long the to do items take". This would affect all of the responses in the current generation's children.

So - the primary claim in the paper is that if we chart the entire tree like an upside-down idea family tree, the most consistent/reliable branches will be the syntax-free (json/yaml/xml formats) vs it's syntax-locked counterparts (nlp, english, spanish, python, java, mandarin, etc).

Like, dramatically, consistently better more consistent, resiliant/resistant to change.

I didn't go to college, so I keep proposing "papers" and he has expressed little interest in doing that, because they usually don't include a data gathering component (other than my experience over the last 20 years actually doing this - and seeing it first hand), those feel more like "white papers". I don't have a good sense of how "research papers" are typically written though, so these words are all relatively foreign to me, but... with an udnerstanding of how ssot.me works, and what the various benefits of having a single source of truth are, the idea that starting with a machine readable, specific, named implementation of the rules not leading to more consistent code vs a nlp english description of the same rules. I mean...

Anyway - before we start designing a specific protocol, what do you think of the idea, in general?

Please do not "blow smoke". Pretend this is my phd research project, and you are my advisor.