CREATING A UNIVERSAL INFORMATION LANGUAGE

OSCAR BENDER-STONE

Abstract. Welkin is a formalized programming language to store information. We introduce its use cases and rigorously define its syntax and semantics. From there, we introduce the bootstrap, making Welkin completely self-contained under a meta-theory based on combinators, equivalent to a provably minimal fragment of arithmetic.

Contents

1. Introduction

Knowledge management is an active problem due to the exponential expansion of new fields. As one major problem, journals are often highly specialized, requring an immense understanding of the broader concepts involved and nomenclature used. This is evident in the sciences, as explored in [1], [2]. Additionally, representing knowledge can be difficult. In mathematics, for example, several attempts have been made to catalog major theories and results. [DESCRIBE ATTEMPTS & LIMITATIONS]. In other subjects, like the social sciences, there are *no* standard terms, and the majority of cited references are books, which are not indexed by many databases [3]. As another challenge, many formats are fragile to incorrect syntax [EXPLAIN AND ELABORATE]. Each of these issues, a small fraction of existing barriers, demonstrate the difficulty in creating a knowledge base with both broad applicability and faithful representations to the original research.

In addition to this [FIRST CLASS OF PROBLEMS NAME], another major hurdle is truth management. [DISCUSS Problems with truth + corrections from papers don't propagate!] What can be done is addressing *information*, the storage of the *asserted* facts themselves, regardless of truth. As one example, suppose a scientist claims, "X is true about Y". One could debate the veracity of that claim, but what we can say is, "This scientist claims, 'X is true about Y'". Even if we doubt that, we could do: "This claim can

be formulated: 'This scientist claims 'X is true about Y'"'. By using these justifications, stating that a claim is expressible, the *syntactic expression* of the claim can be separated from its *semantic truth value*.[1] I will make this more rigorous in later sections, but this means we can build knowledge bases ontop of information systems using flexible extensions.

Information has been extensively studied through *measurements* in Algorithmic Information Theory (AIT). The founding idea of AIT is the Minimum Description Length (MDL) principle, that the best definition for an object is the smallest description that describes it. To formalize this idea, Kolmogorov defined a description as a *program*, and the Kolmogorov complexity of a string as the length of the *smallest program* that computes that string (see [4]). This program is defined via a Turing-computable programming language, and there is a different constant factor depending on the language, but AIT focuses on the asymptotic complexity. A cornerstone of this framework is providing the reason underlying cause of Gödel's incompleteness theorems, Turing's halting problem, Tarski's undefinability of truth, and more: *not all information can be compressed into a finite description.* This view was articulated by Chaitin on information compression. He defined Ω as the probability that a random Turing machine will halt and proved that it cannot be compressed computably. This result is a major theme in AIT, to address the limits of computation.

TODO: REWORD as needed + merge with discussions on FAIR.

In addition to addressing these limits, Chaitin's results have profound consequences for the nature of mathematics. He explains:

> *Mathematics...has infinite complexity, whereas any individual theory would have only finite complexity and could not capture all the richness of the full world of mathematical truth.* [5]

Chaitin's claim extends beyond mathematics; the extent of research areas are so vast that the idea of a *single* theory would fail to faithfully reproduce these disciplines. The study of the areas *themselves* is needed to faithfully represent them. This has been explored in Béziau's field of Universal Logic [6], where the aim is to study *logics* and not a *single* logic. In short, Chaitin's result, and the works in Universal Logic and others, demonstrate that research must be represented *flexibly* as well as faithfully.

The problem with a flexible representation system is precisely *how* to accomplish this. AIT provides asymptotic results on information *measurement*, but does not provide a guide on the fixed representation to use. Chaitin created a LISP variant, designed specifically for the ease of implementation and analysis [7], but this does not address the faithful representations of other languages. Additionally, Universal Logic provides a single definition of a logic, one which can be tedious in exotic logics. Each of these issues underly the importance of *organization* itself, which emerge in the proliferation of general-purpose programming languages.

TODO: Explain all claims concretely but concisely, as well as cite other possibilities (e.g., HoTT for logics)! And add anything

As a concrete example of organizational challenges, consider a Python program in Listing 1.

TODO: Finish transpilation example! Just show we care about *storing* the underlying semantics.

[1] One might be worried about a paradox, such as "This claim is expressible: this claim is not expressible." We will avoid this using a clear separation of the overarching metatheory and object theory, with the former being syntactical in nature. To express this separation, we write quotes around the claim itself.

```c
int main() {
  return 0;
}
```

```python
def main():
  pass

if __init__ == "__main__":
  pass
```

Listing 1. Example programs in C (left) and Python (right).

In light of the persistence of organizational issues, another step is required beyond Chaitin's reasoning: we must consider *universal building blocks* themselves. Sticking to Turing machines is sufficient in AIT and studying asymptotics, but the *actual* storage must take place. This is the motivation and driving force of this thesis.

1.1. Goals.

The aim of this thesis is to create a universal information language to standardize *all* formal representations. I call this language **Welkin**, an old German word meaning cloud [8]. This aim will be made more precise in the later sections, where we will formally define a verifier for a programming language.

- **Goal 1:** universality. This language applies to ANY checker. Needs to be extremely flexible towards this goal, so we can ONLY assume, at most, we are getting a TM as an input, or it's somehow programmed.
- **Goal 2:** standardized. Needs to be rigorously and formally specified.
- **Goal 3:** optimal reuse. With respect to some critierion, enable *as much* reuse on information as possible.
- **Goal 4:** efficiency. Checking if we have enough information *given* a database much be efficient!

1.2. Organization.

- Section 2. Foundations: define the meta-theory used + verifiers. Also outline the Trusted Computing Base.
- Section 3. Information Systems: explore information in the context of verifiers. Then synthesize a definition to satisfy Goal 1.
- Section 4. Information Reuse. Develops the optimal informal system (w.r.t to a metric defined in this system) to satisfy Goal 3.
- Section 5. Syntax: Go over the simple LL(1) grammar, which is similar to JSON and uses python syntax for modules.
- Section 6. Semantics: Defines information graphs and their correspondence with the optimal informal system in Section 3.
- Section 7. Bootstrap. Fulfill Goal 2 with both the Standard AND the complete bootstrap.
- Section ?. Prototype. Time permitting, develop a prototype to showcase the language, implemented in python with a GUI frontend (Qt) and possibly a hand-made LL(1) parser.
- Section 8. Conclusion. Reviews the work done in the previous sections. Then outlines several possible applications.

2. Foundations

2.1. **Abstract Objects/Slates.**

- Brief philosophical discussion
  - Discuss Kit Fine's arbtirary objects.
    * Address symbol groundin problem + circularity: Fine uses FOL to define a notion **used** to construct FOL
    *
  - Emphasize **pragmatism**, echoing intro. It matters how we can **use it** for this language, **not** epistemological statements or certainty (what "is" or "isn't")
- Establish notion of a slate
  - Bring up notion of "tabula rasa"
  - Want a "clean slate" that can be "assigned an interpretation" arbitrarily
    * Make main defense as to why this is universal; need to allow **any** extensions, so need to be arbitrarily imbued by interpreters/oracles.
    * Note: no guarantees on what interpreters there "are" or limiations, e.g., humans have finite lifespans.
    * Main point: to ensure arbitrary interpretation, need clean slates! Argue this is pragmatic (i.e., useful practically)
  - This is completely informal and depends on the interpreter.
  - Idea for formalization: treat **handles/ids** slates as the discrete objects
    * Analogy: sorting through an inventory of dishes. Will connect back to organization!
    * THEN can use formal systems/computable functions around those IDs to make formal claims
    * Why formal systems? Because we want to assert claims! Important pragmatically! Just like keeping track of inventory or specific points! Or being a historian!
    * Use to define information! Expand on how this improves notion of infons
    * Powerful aspect: can shape AROUND new slates! Provide examples in Slate Logic

2.2. **Slate Logic.**

- Definition
  - Define binary strings. Assign these to slates.
    * Have **slate variables**. This is our entry point into arbitrary interpreters.
    * Can change meaning based on interpretation/context! Emphasize how there can be a many to one relationship, and we need to increase formal systems available to distinguish between them!
    * Emphasize need for a function that can enumerate these slate variables. So NOT just one slate variable. Maybe provide lemma on impossibility of doing more (within a formal system?)
  - Combinators
    * Want a **simple** presentation to define theory.

We do require substitution (variables are important here!), but want to present theory with combinators.
    * Emphasize that this is a bootstrap/easier way to start.
  Just like starting somehwere on a map and then relocating (make this clearer!)
- Justifications: what we can assert about **formal objects**
    * Inspired by Artemov's logic of proofs. Will connect back in next two sections with serial consistency!

- Examples
  - Sorting dishes (analogy from before)
  - Map analogy, with places as IDs AND paths
    * Emphasize that new objects can be given IDs arbitrarily; that is why we need (countably) infinitely many IDs!

2.3. **Coherency.**

- Definability
  - Show that this generalizes Padoa definability
    * Classic exapmle where this is used: showing congruence is not definable in terms of betweeness
    * On the reals, $x \mapsto 2x$ is monotonic but does not preserve congruence.
    * In HOL, only one direction shows. Determine how to strengthen to claim to ensure equivalence
  - Basic idea: notion A is definable via notion B iff every map that preserves B also preserves B
  - Want to use this basic idea to talk about information preservation
- Coherency of a System
  - What systems are **useful** or can talk about other systems?
    * Don't want: empty system or one with ALL The rules. These are will have low usefulness
    * Want: complex, intricate structures. Problem is, lots of notions for this! Need to determine a general notion, using slates!
  - Use coherency as basis for information organization

2.4. **Translations Between First Order Logic.**

- Want easy access to first order logic
  - Review literature. Notable examples:
    * SMT solvers in Rocq + Lean (via monomorphization of types)
  - Problem: abstractions are hard to convey! Lots of "bloat"
  - BUT SMT solvers are very well established, particularly with Gödel's completeness theorem.
  - How to get best of both worlds? Solution: slates!
- First step: define extension to first order logic (let's call it, say, FOL(Slate))
  - Add slates as a special sort, but focuses on first order terms.
    * Emphasize that there are FOL theories **weaker** than combinators.

So, with a coherency argument, argue that FOL can be powerful **precisely because** RE is possible, WITH the combination of the completness theorem. (Not possible in all logics!).

- Second step: show that FOL(Slate) is equivalent to FOL by treating slates as an additional sort.
- Straightforward, but emphasize rule on slates on making meanginful/useful abstractions!
- IF time allows, provide experiments, but mostly argue why, based on the argument for slates, this would work.
- Argue that you could AT LEAST embed the necessary abstractions via slates. And organization will help show this is feasible with a theoretical argument (but it's not exponential time. It is (hopefully) ACTUALLY feasible.)
- Final step: show that there is an equivalent embedding that **preserves** slates.
  - Important part: preservation up to iso!
  - Maybe bring up Jose Mesegeru's "epsilon-representation distance" notion

3. Universal Systems

- Provide previous section (translation to FOL) as a major example
- Generalize from the case of a formal system from an earlier draft
  - Earlier definition: (D, R), with D a grammar and R a set of RE rules
  - Universal system: U = (D_U, R_U) is universal if, for each formal system S, there is a term t in D_U such that derivations in S are reflected and preserved via t in D_U. So they are faithfully encoded
  - Earlier proof: a system is universal iff it induces a comptuable, RE full sub-category of the category of formal systems.
  - Refine these ideas to use slates + coherency from before. Can involve more ambitious encodings!
  - Also develop reflection!
- Hint at topic of next section, or smooth out transition. Next section is discussing **which** universal system to use or how to effectively translate between them

4. Information Organization

- Main question: **which** universal system to choose? Is this practical?
  - What is a suitable criterion for a base theory?
  - Recall aim: want to mechanically store systems for a database
    - What if possible performance degredation? Will we get stuck if we start with one architecture? Will we have to adjust later?
    - Aim is to ensure architecture is completely flexible and can automatically adapt
    - One key metric: ability to store as many systems coherently as possible, i.e., store as much information as psosible
  - Main problem: Blum's speedup theorem
    - Briefly generalize this for slate logic
    - Show that no single way to completely organize systems based on a computable metric.

This is part of the need for new search techniques!

* Want to separate search from storage though, but we want to improve

stored results **with** new results. This forms the idea behind the database architecture: have a simple way to store results that automatically gets better with new techniques/results.

* Need explicit proofs for this! Not sure how to store certificates…

4.1. **Impossible Classes.**

The reason to restrict our transformations is two-fold. First, we need to ensure we can *verify* them efficiently. Determining whether a morphism between two formal systems exist can be reduced to the Halting problem, and is therefore not practical for defining an optimal formal system. Second, if we include those tranformations that we *can* effectively check, no optimal formal system exists.

Theorem 4.1.0. *With respect to the class of all computable transformations that can be computably verified, there is no optimal formal system.*

4.2. **Efficient Querying.**

Instead of making proofs most efficient as is, we want to support finding optimal representations. But we want to do this from an efficiently queryable system, which *is* the most optimal.

5. The Welkin Language

5.1. **Syntax.**

- Want to include essential components and use slates

for the rest. May incluce lemmas to show **why** these notions are useful (but make be part of a separate section).

- Scoping: need to avoid collisions! With slates, can expand further and generalize to, e.g., nodes with sharing
- Rewrite rules: need the formal system part to

be clear that is ultimately symbolic.

5.2. **Semantics.**

- Semantics on AST
  - Terms: graphs
  - For ease of use, include a null node

  that is the root of the tree. This represents the module itself.
- For information organization: integrate with previous section
  - Emphasize how this is a useful tool and can ensure

  new information content is being created (at least, that can be distnguished from the current module). If already existing, but that doesn't match the user's expectations, they need to refine it! OR, maybe it **does** match similarly with something else! (e.g., hidden connections between math and music)
- Emphasize pragmatics as well, via slates

6. Bootstrap
    - Provided in bootstrap.welkin file or similar. Main module is welkin, which needs to:
        - Provide slates, so inductive definition of binary strings + variables
        - Explain combinators
        - Explain the basics of universal systems. This is very important!
            * Can elide proofs IF there is enough information content.
            * TODO: figure out how proofs might work in this setting
        - Provides syntax and semantics of Welkin itself
    - This thesis: will prove that the AST generated from this file is correct AND that it does, with a suitable interperation bootstrap itself.
    - Explain how slates expand the envelope for implementations, BUT ensures that the final product, the syntax + semantics checkers and the information organization, can be externally seen!

7. Conclusion
    - Review of thesis
        - Developed slate logic + bi-translation with FOL
        - Developed locally optimal organizational technique that can improve based on annotations/certificates
        - Introduced the language, with a straightforward graph syntax and semantics
        - Builds upon the last section
        - Bootstrapped standard + used coherency condition
    - Significance
        - Backwards AND forwards compatbile standard that bootstraps itself. Easy for implementations!
        - Applications to any human subject
            * Sciences
            * Liberal arts
            * Economics
            * Etc.
    - Future work
        - Programming language semantics + synthesis
            * Incorporate broader aspects + intent of users! ESSENTIAL for new programming languages to be able to discuss pragmatics in some way!
            * Also reproducible AND executable specifications, though creating an engine to execute these is far beyond the scope of the thesis
        - Organizing large corpuses of human text
        - Numerous applications to AI and improving results
            * Emphasize role of symbol grounding problem in AI

References

1. Fanelli, W., Daniele AND Glänzel: Bibliometric Evidence for a Hierarchy of the Sciences. Plos One. 8, 1–11 (2013). https://doi.org/10.1371/journal.pone.0066938

2. Casadevall, A., Fang, F.C.: Specialized Science. Infection and Immunity. 82, 1355–1360 (2014). https://doi.org/10.1128/iai.01530-13
3. Archambault, É., Vignola-Gagné, É., Côté, G., Larivière, V., Gingras, Y.: Benchmarking scientific output in the social sciences and humanities: The limits of existing databases. Scientometrics. 68, 329–342 (2006)
4. Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer Publishing Company, Incorporated (2019)
5. Chaitin, G.: The Limits of Reason. Scientific American. 294, 74–81 (2006). https://doi.org/10.1038/scientificamerican0306-74
6. Béziau, J.-Y.: Universal Logic: Evolution of a Project. Logica Universalis. 12, 1–8 (2018). https://doi.org/10.1007/s11787-018-0194-7
7. Chaitin, G.: Elegant Lisp Programs. (2003). https://doi.org/10.1007/978-1-4471-0015-7_2
8. Dictionary, O.E.: welkin, n., https://www.oed.com/dictionary/welkin_n

Department of Mathematics, University of Colorado at Boulder, Boulder, CO
Email address: oscar-bender-stone@protonmail.com