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 depth and breadth of fields. Journals are often specialized, requring an immense understanding of the broader concepts involved and nomenclature used. This is evident in the sciences, as explored in [1] and [2]. Additionally, creating common representations shared accross subdisciplines can be difficult. In mathematics, for example, several attempts have been made to catalog major theories and results.

. 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]. The problems posed by broadly and faithfully capturing subjects demonstrate the herculean task of scalable knowlege management.

TODO: DESCRIBE ATTEMPTS & LIMITATIONS

In attempt to address these challenges, several solutions have been proposed, but none completely fix these issues.

The aim of this thesis is to create a universal information language to standardize *any* knowledge representation. I call this language **Welkin**, based on an old German word meaning cloud [4]. This language has the following goals.

1.1. Goals.

- **Goal 1: Universality.** The language must include unspecified, user created parameters to accomodate for arbitrary concepts and ideas.
- **Goal 2: Standardization.** The language needs a rigorous and formal specification. Moreover, the bootstrap must be formalized, as well as an abstract machine model. The grammar and bootstrap must be fixed to ensure complete forwards and backwards compatbility.
- **Goal 3: Efficiency.** Local queries in the database, determining if there is enough "explicit" information, must be efficient.

1.2. Organization.

- Section 2. Foundations: defines the meta-theory use and its connections to first-order logic and fragments of arithmetic.
- Section 3. Information Organization. Develops the optimal informal system (w.r.t to a metric defined in this system) to satisfy Goal 3.
- Section 4. Defines the syntax and semantics.
- Section 5. Bootstraps Welkin by proving that there is a Welkin node that contains enough information about the standard. Fulfills 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

This section establishes the meta-theory for Welkin, capturing the major components of symbolic systems. The meta-theory consists of two interconnected parts:

- The computational part, which can express any partial computable function via combinators.
- The logical part, which enables checking on properties of symbols and certificates on normal forms.

This approach generalizes Fefermans' Explicit Mathematics framework [5], which intentionally separates computation from logic. We build upon this framework by incorporating slates, as well as having a robust model of trust through work done by Artemov [6].

2.1. Minimal Trusted Computing Base.

We first need to establish the logical machinery underlying the theory.

2.2. Philosophy Behind Slates.

Slates are inspired by several theories. The most prominent arises in Burgin's theory of information [7], [8]. Given an informatoin system $*\,R\,*$, he specifies an infological system IF$(*\,R\,*)$ unspecified parameter, meant as a way to tailor the system to domain specific terminology. However, Burgin does not develop rules to work with these parameters, and the rest of his results exacerbate a divide between these and the formal semantics.

TODO: Finish writing up this paragraph + ensure accuracy!

Seeking to augment Burgin's approach, we examine Kit Fine's notion of arbtirary objects [9]. In his seminal work, he formalizes Cantor's definition of numbers via "Cantorian abstraction". Instead of defining a number through a Fregean set of all things with that quantity, or an artifical construct like Von Neumann ordinals, Cantor defined a number as a set of abstract entities. Fine proceeds to formalize this within a first-order logic. Several papers have explored this in different ways, including [10]. However, Valaris [11] provides a key objection to using induction... We address this with a powerful symbolic rule for slates that make slates themselves rules, in a specific sense.

2.3. **Slate Logic.**

TODO: Write up details in next section! Make sure this is accurate!

- 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!

2.4. **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

2.5. **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. Dictionary, O.E.: welkin, n., https://www.oed.com/dictionary/welkin_n
5. Feferman, S.: Arithmetization of metamathematics in a general setting. Fundamenta Mathematicae. 49, 35–92 (1960)
6. Artemov, S.: Serial properties, selector proofs and the provability of consistency. Journal of Logic and Computation. 35, exae34 (2024). https://doi.org/10.1093/logcom/exae034
7. Burgin, M.: Foundations of Information Theory, https://arxiv.org/abs/0808.0768
8. Burgin, M.: Theory of Information. World Scientific (2009)
9. Fine, K.: Reasoning with Arbitrary Objects. Blackwell, New York, NY, USA (1985)
10. Shapiro, S.C.: A logic of arbitrary and indefinite objects. In: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning. pp. 565–575. AAAI Press, Whistler, British Columbia, Canada (2004)
11. Valaris, M.: Induction, Normality and Reasoning with Arbitrary Objects. Ratio. 30, 137–148 (2017). https://doi.org/https://doi.org/10.1111/rati.12129

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