# 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 the meta-theory of Goedel's System T (equi-consistent to Peano Arithmetic).

CONTENTS



## 1. INTRODUCTION

- Engineering Research has empowered humanity for centuries.
  - Provide examples (mechanics, steam engine, medicine, transistors, etc).
  - Cite history of research, with STEM in mind. (Maybe mention liberal arts?).
- Research *communities* are extremely diverse.
  - Examples: many concentrations in CS, Math, etc.
  - Discuss pervasive issues in *bridging* research.
    * Different languages, approaches, journals, even subtly distinct definitions!
    * Maybe cite original example of BSM + UDGs, in which combinatorialists

  operate separately from SMT-LIB, even though both can support the other.

    * Another example: My Cryptographic professor and

  a CS researcher that has made a cryptographic tool are not aware of each other!

- My realization: lots of *potential* for collaboration, BUT can be difficult. Cite Madhusdan P. in my first meeting with him, in which he said that PL communities are separate for a reason.
    - Also cite work on Universal Logic; discuss history of not a *single* logic prevailing, but instead many. Same thing with structures, computational models, and other mathematical objects.
    - Mention Meseguer's work with rewriting logic and how representations can be difficult! Also outline potential missing elements in rewriting logic.
- Develop the idea of a *framework*: shift from a *universal theory* to *universal "building blocks"*
  - Original idea (pre-20th century): there is a *single* theory/object we must mold everything else into. (Draw a picture!)
  - Shaken up by the foundational crises in the 20th century, but also in *every* field in different ways. Physics with quantum mechanics, biology with diversity of organisms, etc.
  - Highlight: communities NEED to keep their notions! This is a *strength* - people have done great things with this philosophy!
  - New idea: *every* formal object is *made* of the same content, but in different ways.
    - This allows communities to shape their own notions *differently*, but this can be compared precisely because they are made of the *same* things. Highlight an analogy with atoms.
    - My aim: show that *information* is precisely this building block, restricted to a computable setting.
- Review past results in Information theory. Possibly expand upon in a separate section.
  - Main theories:
    - Shannon + entropy:
      - Main founder of information theory
      - Information = "reduced uncertainty"
      - primarily probabilistic and based on *communication of bits*. Not enough for *semantics*!
      - Takeaway: information is *used* in communication
    - Kolomogorov + complexity:
      - Minimum Description Length (MDL): we should describe objects with the smallest description possible.
      - Kolmogorov complexity = length of *smallest* program accepting a string
      - Practical problem: not computable!
      - Bigger problem: *measures* information, but does not *define it*
      - Takeaway: provides a *computational* lens for information
    - Scott domains:
      - Introduced information systems!
      - Problem: divergence from . Also divergence from information in other senses, like ontology (OWL, etc.)

        * Ontologies:
            . Frameworks: OWL, Conceptual Graphs, etc.
            . Problem: pretty restricted! Might only be first-order.
- Synthesis of information theory + past paragraph
    - We want to explore *information* as a universal framework.
        * Major goal: address information *on* information.
            . Motivation: transfer between different things!
                - Logics: proofs!
                - Models: properties!
                - Solvers: techniques and checkpoints! Reduces computation!
                - And more!
    - But what *is* information?
        * We'll exclude less tangible notions, like perception or emotions.
        * Ultimately, even in an informal setting,

    will need to store (tangible) information as a computable string!
        * So many examples: hard to know where we should go!
            . Algorithmic ideas.
            . Specific formulations in logics.
            . High level properties.
            . Probabilistic information.
        * Need to do some detective work - that's part of this thesis!
        * *Can* start at an indirect approach to ensure we don't

    miss anything: indirectly define things by how they are *checked*. This checker/verifier MUST be a Turing machine (using a standard notion!).
        * Emphasize: finding a certificate is *not* guaranteed, e.g., finding a proof of a theorem in first order logic.
        * To simplify this: we have *verifiers/checkers* that take in binary strings called *certificates*. Correct certificates are accepted by the checker, and rejected otherwise.
        * *Use this definition to justify universality!*
- Aim of thesis: create a universal information language to store information in a standardized way.
    - Goal 1: universality. This language applies to ANY checker.
    - 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!
- Organization (Maybe provide as another table *with* descriptions?)
    - Section 2. Base Notions: define the meta-theory used + verifiers.
    - Section 3. .
    - Section 4.

## 2. Foundations

We introduce the base theory needed for this thesis. Our work builds on deep inference, developed by Strassburger [1]. and many others. We formally define a formal system and then proceed to show this can be encompassed in a deep inference framework. These sections are closely replicated as steps in the bootstrap (see Section 10).

We will keep this self-contained; additional references will be provided in each subsection. For general notation, we write := to mean "defined as".

### 2.1. Base Notions.

Before continuing, we must introduce some fundamental notions. We introduce **alphabets**, using three columns: the first is the symbol name, in monospace font; the second is the mathematical notation used; and the third is the symbol's name. See Table 1 for the template. Note that we informally use natural numbers. However, each definition is self-contained. See Remark 2.1.1 for a related discussion. Additionally, sometimes our symbols may be *multiple* tokens; this is addressed in Section 8.

| | | Symbol | Notation | Name |
|---|---|---|---|---|
| $\mathcal{A}$ | := | `s0` | $s_0$ | **symbol zero** |
| | \| | `s1` | $s_1$ | **symbol one** |
| | \| | | $\dots$ | |
| | \| | `sn` | $s_n$ | **symbol $n$** |

TABLE 1. Template for an alphabet $A$.

**Definition 2.1.0.** The **binary digits (bits)** are given by:

| | | Symbol | Notation | Name |
|---|---|---|---|---|
| Bit | := | `0` | 0 | **zero** |
| | \| | `1` | 1 | **one** |

TABLE 2. The symbols used in bits.

Recursive definitions are given in the form of a **judgement** (Figure 1), consisting of **premises** on top and a **conclusion** on the bottom.

$$\frac{P}{C}\ J$$

FIGURE 1. Template for a judgement.

**Definition 2.1.1.** The **language of words** $\mathcal{L}_W$ is provided in Table 3. A **word** $w \in W$ is given by the judgements in Definition 2.1.2.

| | | Symbol | Notation | Name |
|---|---|---|---|---|
| $\mathcal{L}_W$ | := | `Bit` | Bit | **See Definition 2.1.1** |
| | \| | `{}` | $\varepsilon$ | **Empty word** |
| | \| | `.` | . | **Concatenation** |
| | \| | `=` | $=$ | **Equality** |
| | \| | `!=` | $\neq$ | **Inequality** |

TABLE 3. Language of words

$$\frac{}{\varepsilon \in W}\ \text{Empty} \quad \frac{w \in W}{w.0 \in W}\ \text{Zero} \quad \frac{w \in W}{w.1 \in W}\ \text{One}$$

FIGURE 2. Recursive definition of words.

*Remark* 2.1.2. The definition for binary strings, as the remaining recursive definitions, serves as a suitable *uniform* abstraction for data. From a physical viewpoint, we cannot *verify* each finite string, a phenomenon related to the notion of "Kripkenstein" [2]. However, we *can* provide the template and is more suitable as a definition, and we presume these definitions are completely contained (i.e., binary strings are defined by a finite combination of *only* the rules above). On the other hand, proof checking will be done in an ultra-finitistic setting and is addressed in Section 10.

Natively, our encoding uses binary. But to simplify this notation, we introduce shorthands using two other number systems.

### 2.2. **System T.**

## 3. FORMAL REASONING

Now with our meta-theory in Section 2, we can proceed to discuss formal systems.

### 3.1. **Formal Systems.**

To bridge information graphs with formal reasoning, We must first define formal systems generally. Our definition is based on three sources:

- Mendelson [3].
- Cook and Reckhow [4] with "formal proof systems".
- Strassburger [5].

**Definition 3.1.0.** A **formal system** is a pair $(\mathcal{F}, \mathcal{R})$ consisting of:

- **formulas** $\mathcal{F}$, a decidable set of binary strings.
- a set of **derivation rules** $\mathcal{R} \subseteq \mathcal{F} \times \mathcal{F}$. We define the **derivation relation** $\Rightarrow_{\mathcal{R}}$ to be the reflexive, transitive closure of $\mathcal{R}$. Furthermore, we require that $\Rightarrow_{\mathcal{R}}$ has a polynomial time verifier $V_{\mathcal{R}}$.

Note that the first condition on $\mathcal{F}$ is redundant: reflexivity in $\mathcal{R}$ ensures that each formula can be recognized in polynomial-time.

**Definition 3.1.1.** Let $\mathcal{S} \equiv (\mathcal{F}, \mathcal{R})$ be a formal system. A **derivation** or **proof** is a sequence of derivation rules. The **category of proofs** $\mathrm{Proof}(\mathcal{S})$ consists of:

- **Objects:** formulas.
- **Morphisms:** proofs between formulas. Concatenation is defined by concatenating sequences.

*Remark* 3.1.2. Strassburger [5] advocates to *define* a logic as a category. But this is not immediate for certain logics. For instance, in the sequence calculus, composition of two proofs is not uniquely defined. Our definition approaches this by using an artificial, inefficient representation. Strassburger's work on deep inference addresses this problem, but instead of a generalization, we intrepret it as an *optimization*; see Section 5.

**Definition 3.1.3.** Let $(\mathcal{F}, \mathcal{R})$ be a formal system, and let $\mathcal{T}$ be a set of formulas. The **deductive closure** of $\mathcal{T}$ is $\mathrm{Th}(\mathcal{T}) = \{\varphi \in \mathcal{F} \mid \exists \psi \in \mathcal{T}. \psi \vdash_{\mathcal{R}} \varphi\}$. We call $\mathcal{T}$ a **theory** if $\mathcal{T} = \mathrm{Th}(\mathcal{T})$. A set of formulas $\mathcal{A}$ serve as **axioms** for a theory $\mathcal{T}$ if $\mathcal{T} = \mathrm{Th}(\mathcal{A})$.

### 3.2. **Universal Systems.**

We want to study formal systems in general. A key invariant we want to preserve is *faithful representations*, or the notion of "$\varepsilon$-representatoin distance" from José Meseguer [6]. The idea is that a *good* representation of a mathematical object is one which preserves and reflects isomorphism. We can treat this as *soundness* (isomorphic representations have actually isomorphic objects) and *completeness* (isomorphib objects produce isomorphic representations) of the representation, respectively. To make this precise,we introduce **transformations**, which are mappings between formulas of two systems, and then proceed to **morphisms**, which are structure preserving maps.

**Definition 3.2.4.** Let $(\mathcal{F}_1, \mathcal{R}_1), (\mathcal{F}_2, \mathcal{R}_2)$ be formal systems. Then a **transformation** $f : (\mathcal{F}_1, \mathcal{R}_1) \to (\mathcal{F}_2, \mathcal{R}_2)$ is a pair $(F, R)$, where $F : \mathcal{F}_1 \to \mathcal{F}_2$ is computable and $R \subseteq \mathcal{R}_1 \times \mathcal{R}_2$ is left-total and if $F(\varphi) = \psi$, then $\varphi R \psi$.

**Definition 3.2.5.** A **morphism** $f \equiv (F, R)$ is a transformation such that $\Rightarrow_R$ is functional. More explicitly, $\varphi \Rightarrow_{\mathcal{R}_1} \psi \Rightarrow F(\varphi) \Rightarrow_{\mathcal{R}_2} F(\psi)$. An **isomorphism** is an invertible morphism whose inverse is also a morphism.

**Definition 3.2.6.** The **category of formal systems** $\mathbb{F}$ consists of:

- **Objects:** formal systems.
- **Morphisms:** defined in Definition 3.2.7.

Note that this algebraic structure satisfies reflexivity and existence of composites.

**Definition 3.2.7.** A **sub-category** $\mathbb{F}'$ of $\mathbb{F}$ consists of a subset of objects and a subset of morphisms. A sub-category is **full** if it removes no morphisms. A **framework** is a subcategory equivalent to $\mathbb{F}$ such that the objects form a decidable set.

Frameworks closely relate to the notion of **universal** formal systems.

**Definition 3.2.8.** A formal system $\mathcal{U} \equiv (\mathcal{F}_{\mathcal{U}}, \mathcal{R}_{\mathcal{U}})$ is **universal** if there is a computable family of injective functions $G = \{G_{\mathcal{S}} : \mathcal{F}_{\mathcal{S}} \to \mathcal{F}_{\mathcal{U}} \mid \mathcal{S} \equiv (\mathcal{F}, \mathcal{R}) \in \mathbb{F}\}$ over all formal systems such that at each fixed system $\mathcal{S}$ and for all formulas $\varphi, \psi \in \mathcal{F}$, $\varphi \Rightarrow_{\mathcal{R}} \psi \Leftrightarrow G_{\mathcal{S}}(\varphi) \Rightarrow_{\mathcal{R}_{\mathcal{U}}} G_{\mathcal{S}}(\psi)$.

Our motivation for defining universal systems is a property called **reflection**, similar to the one outlined in [6]. That is, universal systems *themselves* can be studied in the context of a single universal system. This enables meta-theoretic reasoning.

**Theorem 3.2.9.** *Every universal formal system induces a framework $\mathbb{F}'$, as the image of the functor $\mathcal{G} : \mathbb{F} \to \mathbb{F}'$, given by $\mathcal{G}(\mathcal{S}) = \left(\mathrm{Image}(G_{\mathcal{S}}), \mathcal{R}_{\mathcal{U}} \cap \mathrm{Image}(G_{\mathcal{S}})^2\right)$. Conversely, every framework induces a universal formal system.*

*Proof.* We must show that $\mathbb{F}'$ is a framework for $\mathbb{F}$. Clearly this is a computable sub-category. To prove $\mathcal{G}$ is an equivalence, notice that $\mathcal{G}$ is full and faithful as a full sub-category of $\mathbb{F}$. Additionally, $\mathcal{G}$ is essentially surjective precisely by construction. This completes the forwards direction.

Conversely, a univeral framework can be formed from a system by creating a computable encoding of the formulas and rules of a system. The family $G$ can then be defined from an equivalence from $\mathbb{F}$ to $\mathbb{F}'$, which can be easily verified to preserve and reflection derivations. $\square$

**Theorem 3.2.10.** *Let $\mathcal{U}$ be a unviersal system. Then for every formal system $\mathcal{S}$,* $\mathrm{Proof}(\mathcal{S})$ *is equivalent to a subcategory of* $\mathrm{Proof}(\mathcal{U})$.

## 4. Information Graphs

We now define the universal framework for analyzing information. Our approach describes information as a *relation*. We show that this framework encompasses both ontology and formal systems.

From there, we analyze the notion of *compressing* information in an effectively realizable way. We establish two key constructions and generalize them as *information transformations*. The main result in this section is showing that the best *effectively realizable compression scheme* can be obtained by appealing to Tarski's fixed-point theorem on the lattice of these transformations.

### 4.1. **Motivating Examples.**

Our motto is this: *information is a relation*. We will consider several examples.

**Example 4.1.0.** Alice tells Bob that "I have a cat". This is a relation that relates Alice to some cat. In contrast, "a cat" is not a relation, and therefore not information.

**Example 4.1.1.** A statement like $2 + 2 = 4$ is information. But it is also *true* information, or *knowledge*. We allow $2 + 2 = 5$ to be information as well, because it asserts *some* relation between $2 + 2$ and 5. A non-example is simply the number 2 or a random binary string `0b010001`. Neither of these are relations because they are missing an *explicit connection*. Note that these relations can be unary, such as $2 = 2$.

**Example 4.1.2.** Information can include quantifiers: "Joe has at least one egg." This asserts the existence of *some* egg. Similarly, we can . However, we want to include more general notions. For instance, we could have quantifiers on *surfaces*: "This ball is red everywhere". This is not builtin directly as a first-order quantifier. Additionally, we would like to allow for modal sentences, like "There is necesarially one marble in the bag" or "There is possibly one marble in the basket". We will consider these generalized connectives into our definition (see Definition 4.2.15).

However, we quickly run into philosophical blockades when we want to *use* information.

**Experiment 4.1.3.** Suppose a person describes their feelings through a painting. Does this painting *convey* information? Perhaps we can infer some emotions, such as feelings of sadness in a rainy scene or happiness in a cheerful one. How exactly do we *use* or even *store* this information? Is this data *subjectively* information, requiring a person as an observer?

We avoid these ideas by focusing on *formal* information. This can be rigorously defined into two key components: a **hierarchy** and a set of **connections**. Our notion is based on **bigraphs**, a data structure created by Robin Milner [7].

### 4.2. **Formal Approach.**

**Definition 4.2.4. Information** is a **bigraph**, a triple $(X, T_X, G_X)$ where:

- $X = V \cup T$ is the **domain**, where $V$ is the set of **variables** and $T$ is the set of **bound terms**, each of which are countable sets of binary strings. We call $\mathcal{P}(X)$ the set of **nodes**.
- $T_X$ is the **place graph** or **hierarchy**, a tree with nodes in $\mathcal{P}(X) \cup \{\bot\}$, where the root is a distinguished element $\bot \notin X$. For nodes $A, B$ we write $A \leq B$ if $A = B$ or $A$ is a descendant of $B$.
- $G_X \subseteq \mathcal{P}(X) \times \mathcal{P}(X) \times \mathcal{P}(X)$ is the **link graph**. We write $(A, B, C) \in G_X$ as $\vdash A - B \to C$. In the case where $B = \emptyset$, we simply write $\vdash A \to C$. Additionally, we write $\vdash A = B$ iff $A \to B$ and $B \to A$.

A **pattern** $P$ is a node such that for some $P' \leq P$, $P' \in V$.[1]

*Remark* 4.2.5. Our notion of bigraph diverges from Milner [7] in several important ways. Firstly, Milner's theory focuses around modeling non-deterministic operations in programming languages. Briefly, he considers place graphs which are *forests* (with special regions), and allows for "holes" in the link graph. We simplify his definition by using a tree (with a designated root $\bot$), and incorporating holes instead as patterns. Secondly, Milner's approach defines an algebra for bigraphs, as well as dynamic semantics (via *bigraphical reactive systems*). This is not immediately natural for our generalized setting; we will return to this issue in information Section 5.

We interpret the elements of $X$ as *parts*, and think of the tree $T_X$ as defining a *part-whole* relation. A key design of Welkin is to enable *multiple* notions of part-hood

[1]Our terminology is adapted from Grigore Roșu's *matching logic* [8]. We will return to this comparison later.

and seamlessly work among these. We provide a motivating example one possible construction.

**Example 4.2.6.**

- **Physical Composition:** Let $X_1 = \{\text{house}, \text{wall}, \text{floor}\}$ and suppose wall and floor are parts of house. In this case, parthood means *physical composition*.
- **Classification:** Let $X_2 = \{\text{animal}, \text{dog}, \text{bird}\}$, and let dog and bird be parts of animal. Parthood is treated as a *taxonomy* among things, specifically animals.
- **Hybrid:** Suppose we want to put $X_1$ and $X_2$ above into a knowledge base. One way to distinguish between *physical composition* and *taxonomy* is to introduce new links: makes and isa, respectively. We take $T_3 = T_1 \cup T_2$ and introduce new relations: wall $\overset{\text{makes}}{\to}$ floor $\overset{\text{makes}}{\to}$ house dog $\overset{\text{isa}}{\to}$ animal, bird $\overset{\text{isa}}{\to}$ animal, respectively.

**Definition 4.2.7.** The **consequence operator** $\vdash$ of a bigraph $(X, T, G)$ is defined by the judgements in Figure 3.

$$\frac{}{\vdash A \to A}\ \text{Refl} \qquad \frac{\vdash A \to B, B \to C}{\vdash A \to C} \qquad \frac{\vdash A \to B, A = A', B = B'}{\vdash A' \to B'}\ \text{Equality}$$

$$\frac{\vdash A_1 \to B_1, ..., A_n \to B_n}{\vdash \{A_1, ..., A_n\} \to \{B_1, .., B_n\}}\ \text{Group}$$

FIGURE 3. Conditions on the link graph, adapted from Meseguer [6].

## 5. INFORMATION COMPRESSION

A natural question arises with universal formal systems: *which* one do we choose? While we have reflection, what is the criterion for the *base* theory? Can this be done? One loose, but natural, mteric is this: *a universal system which stores as many "interesting" proofs as possible*. The motivation behind this metric is to enable effective querying of "good" proofs.

We will show that, under a restricted notion of transformation, there is an optimal universal system. This will form the encoding under Section 9 and provide a justification for Welkin as this base theory.

## 6. 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 6.0.** *With respect to the class of all computable transformations that can be computably verified, there is no optimal formal system.*

## 7. 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.

## 8. Syntax

## 9. Semantics

### 9.1. **Terms.**

We expand upon to anaylze the ASTs generated from Section 8.

## 10. Bootstrap

## 11. Conclusion

## References


1. Strassburger, L.: From Deep Inference to Proof Nets via Cut Elimination. Journal of Logic and Computation. 21, 589–624 (2011). https://doi.org/10.1093/logcom/exp047
2. Loar, B., Kripke, S.A.: Wittgenstein on Rules and Private Language. Noûs. 19, 273 (1985)
3. Mendelson, E.: Introduction to Mathematical Logic. Chapman & Hall/CRC (2009)
4. Cook, S.A., Reckhow, R.A.: The Relative Efficiency of Propositional Proof Systems. Logic, Automata, and Computational Complexity. (1979)
5. Straßburger, L.: What is a Logic, and What is a Proof?. In: Beziau, J.-Y. (ed.) Logica Universalis. pp. 135–145. Birkhäuser Basel, Basel (2005)
6. Meseguer, J.: Twenty years of rewriting logic. The Journal of Logic and Algebraic Programming. 81, 721–781 (2012). https://doi.org/https://doi.org/10.1016/j.jlap.2012.06.003
7. Milner, R.: Bigraphical Reactive Systems. In: Larsen, K.G. and Nielsen, M. (eds.) CONCUR 2001 — Concurrency Theory. pp. 16–35. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
8. Rosu, G.: Matching Logic. Logical Methods in Computer Science. (2017). https://doi.org/10.23638/lmcs-13(4:28)2017


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