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

Information Management (IM) is an open area of research as a result of the depth and breadth of disciplines. In terms of depth, many areas are often specialized, requiring an immense understanding of the broader concepts involved and nomenclature used. This specialization is evident in the sciences, as explored in [FG13], [CF14]. Additionally, in terms of breadth, creating common representations shared across sub-disciplines can be difficult. For example, mathematics has extremely diverse disciplines, and connecting these areas is an open problem in scalability [Car+21]. Moreover, creating a standardized form across communities is challenging. In other subjects, like the social sciences, there are no standard terms [Arc+06], and in the humanities, representing certain artifacts as data is involved [Har+20]. More broadly, IM *itself* is divided from distinct approaches that lack interoperability [AJ23]. Certain frameworks equate IM to Knowledge Management (KM) and assert that information must be true [Edw22]. These problems, in both faithfully and broadly storing information, demonstrate the enormous task of effective IM.

In response to these challenges, several solutions have been proposed, but none have been fully successful. In the sciences, a group of researchers created the Findable Accessible Interoperable Resuable (FAIR) guidelines [Wil+16]. Instead of providing a concrete specification or implementation, FAIR provides best practices for storing scientific information. However, multiple papers have outlined problems with these overarching principles, including missing checks on data quality [Gui+25], missing expressiveness for ethics frameworks [Car+21], and severe ambiguities that affect implementations [Jac+20]. Along with the sciences, there are several projects for storing mathematical information (see [CF09] for more details). Older proposals, including the QED Manifesto [KR16] and the Module system for Mathematical Theories (MMT), aimed to be more general and have seen limited success. More centralized systems, like `mathlib` in the Lean proof assistant [The20], have seen adoption but do not give equal coverage nor are interoperable with other systems. Beyond more "hard" fields, IM in the humanities has few models, including aan adaption of FAIR [Har+20] and discipline specific, linked databases in the PARNTHEOS project [Hed+19]. Each of these proposals, even within speciic fields, fail to accommodate for all of the mentioned challenges.

In addition to domain specific proposals, there are approaches for general IM which still fail to resolve all issues. One prominent example is Burgin's theory of information [Bur09] that comprehensively includes many separate areas for IM, including the complexity-based Algorithmic Information Theory (AIT), through a free parameter called an "infological system", which encompasses domain specific terminology and concepts. In contrast to other approaches, Burgin's generalized theory is flexible and enables greater coverage of different kinds of information [Mik23]. Despite this coverage Burgin does not closely tie the free parameter with his formal analysis of AIT, making it unclear how to use this in a practical implementation. Broad frameworks for IM, along with the specific proposals, have severe shortcomings, highlighting major obstacles for IM.

This thesis introduces a language to resolve these issues. I call this language **Welkin**, based on an old German word meaning cloud [Dic25]. The core result of this thesis is proving that Welkin satifies three goals: is **universal**, **scalable**, and **standardized**. For details, see Table 1. The core idea is to generalize Burgin's free parameter and enable arbitrary representations in the theory, controlled by a computable system. The notion of representation builds on Peirce's semiotics, or the study of the relationship between a symbol, the object it represents, and the interpreter or interpretation that provides it that meaning [Atk23]. Moreover, to address queries on the validity of truth, we use a relative notion that includes a context managed by a formal system. Truth can then be determined on an individual basis, providing flexibility to any discipline. The focus then shifts to the usefulness of representaitons based on a topological notion of how "foldable" a structure is, which we call **coherency**. This approach is inspired by coherentism, a philosophical position that states truth is determined in comparison to other truths. [Bra23]. We incorporate ideas from coherentism to identify which representations identify their corresponding objects, and we define information as an invariant under these coherent representations. We include definitions on a *working* basis as what is most practical, not an epistemological stance that can be further clarified in truth systems. Additionally, we keep the theory as simple as possible to make scalability and standardization straight-forward.

Goal 1	**Universality**	The language must enable any user created parameters, whose symbolic representation is accepted a computable function. Every computable function must be definable in the language.
Goal 2	**Scalability**	The database must appropriately scale to broad representations of information. Local queries must be efficient. Certificates must be available to prove cases where optimal representations have been achieved.
Goal 3	**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.

TABLE 1. Goals for the Welkin language.

This thesis is organized according to Table 2.

Section 2	**Motivating Example**	Introduces a high-level example, with geographic maps, to explain the core concept in Welkin.
Section 3	**Foundations**	Explains the meta-theory.
Section 4	**Syntax**	Provides the grammar and proof that it is unambiguous.
Section 5	**Semantics**	Explains how ASTs are validated and processed. Develops representations and coherency, and connects these to a working definition of information.
Section 6	**Information Organization**	Develops a Greedy algorithm to locally optimize information. Creates a certificate that demonstrates when a representation is optimal relative to the current information database.
Section 7	**Bootstrap**	Bootstraps the language.
Section 8	**Conclusion**	Concludes with possible applications, particularly in programming languages and broader academic knowledge management.

TABLE 2. Organization for the thesis.

2. Motivating Example

We illustrate Welkin with a motivating example: geographic maps.

Fix some landscape L. A map provides a representation to guide travelers in L, usually through coordinates and directions. Some common elements include landmarks, paths, and regions.

There are two major problems in creating "good" representations:

1) Between two representations, how can we tell they represent the same entity?
2) Given a representation that represents some referant, how can we distinguish from other possible referants?

In the context of maps, we can make these problems more concrete:

1) Consider two maps M, M'. How can we tell whether some landmark O in M represents the same entity as O' in M'?

2) Consider a map M, and suppose there are landscapes L, L'. With the goal to have M represent L, how does M distnguish between L and L'?

2.1. **Scenario 1: Relating Distinct Representations.**

2.2. **Scenario 2: Distinguishing Referants.**

This overarching example demonstrates how two sources communicate about some entity, or how a source's representation can distinguish between two entities.

3. Foundations

To introduce our foundations, we need to ensure the language is *expressive* enough. As an information language, the core design is to mechanize the storage and retrieval of information, so we generally say that this must be processed as a computable function, based on general consensus. Thus, at the very least, we must express all computable function; this is shown in Theorem 5.3.4.

However, we need more than computable functions: we seek *clarity* in concepts. We need to include meaning *with* the symbols, so we at least need representations. This encompasses partial computable functions as well by modelling non-termination as a unit, as well as Input/Output mechanisms.

TODO: Prove the claim for partial computable functions and IO!

Given these two components, managing information with computable functions and including representations, we argue that Welkin precisely captures anything *describable by a computable representation*. Practically, we impose defining things based on *restrictions*. As a justification, suppose we abstractly consider representing *anything*. The problem is, this is an open concept, and we may easily miss certain notions. We could also say there are *no* restrictions, but this, too, is similarly problematic. But the least *practical* restriction is precisely having representations accepted by a computable function. This provides a best of both worlds: the flexibility for any (reasonable concept), and guarantees mechanically feasible operations on representations.

Now, a key component of this argument, as well as our truth management system, is proving *true* things about computable functions. We develop the machinery through Welkin's meta-theory. We keep this section self-contained with explicit alphabets and explicit recursive definitions. For simplicity, we will use the notation $a_0, \ldots, a_n$ for a finite list of items. We will revisit the notion "finite" more rigorously in Section 7.

3.1. **Words.**

Definition 3.1.0. The **alphabet of binary words** is $\mathcal{A}_{\text{bword}} ::= \text{bit} \mid . \mid w$, where $\text{bit} ::= 0 \mid 1$. A **binary word** is defined recursively: the symbols 0 or 1 are strings, or if w is a string, then so are $w.0$ and $w.1$. We set concatenation to be right-associative, i.e., $(w.w').w'' = w.(w'.w'')$, and safely abbreviate $w.w'$ as ww'. We write $w \in \text{bword}$ to denote that w is a binary word.

For simplicity, we extend the alphabet to include two common bases: decimal and hexadecimal.

Definition 3.1.1. The **alphabet of words** $\mathcal{A}_{\text{word}} ::= \mathcal{A}_{\text{bword}} + \mathcal{A}_{\text{dec}} + \mathcal{A}_{\text{hex}}$, where:

- $\mathcal{A}_{\text{dec}} ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
- $\mathcal{A}_{\text{hex}} ::= A \mid B \mid C \mid D \mid E \mid F$

A **word** is a concatenation of either only binary digits, only decimal digits, or hexadecimal digits. Each of these are denoted with different prefixes: decimal has none, binary uses $0b$, and hexadecimal uses $0x$.

Using binary words simplifies the bootstrap, so while these digits are included, they are *defined* in terms of binary words, see Section 7. Additionally, we will frequently use **bytes**, which are eight bits, or equivalently two hexadecimal digits.

3.2. **Arithmetic.**

4. Syntax

Now, the base encoding for Welkin is in US-ASCII, formally defined below.

4.1. **Encoding.**

Definition 4.1.0. US-ASCII consists of 256 symbols, listed in Table 3.

Dec.	Hex.	Glyph	Dec.	Hex.	Glyph	Dec.	Hex.	Glyph	Dec.	Hex.	Glyph
0	00	NUL	32	20	Space	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Table 3. US-ASCII codes and glyphs.

4.2. **Strings.**

We reserve the term **string** when a word is explicitly enclosed in deilmiters, namely single or double quotes. The precise definition is involved, due to including quotes within a string, which are called “escaped quotes”. To detect escaped quotes, we use our fixed set of characters (see Table 3).

Definition 4.2.1. A **single-quoted string** is defined recursively.

The definition of double-quoted string is analogous.

4.3. **Grammar.**

Definition 4.3.2. Backus-Naur Form (BNF) consists of productions. Writing $r \coloneqq a_1 \mid \ldots \mid a_n$ is shorthand for the rules $r \coloneqq a_1, \ldots, r \coloneqq a_n$. A **derivation** is a sequence of steps, recursively defined by starting with the empty derivation, and if d is a derivation and s is a step, then $d.s$ is a derivation. We write $\alpha \overset{*}{\Rightarrow} \beta$ if there is a derivation from α to β.

Now, we formalize an unambiguous form of EBNF for our use case.

Welkin’s grammar is displayed in Listing 1, inspired by a minimal, C-style syntax. Note that the empty string is not accepted, but is instead represented by the string `{}`.

```
start ::= (term ",")* term
term ::= arc | graph | base
arc ::= (term "-" term "->")+ term
      | (term "<-" term "-")+ term
      | (term "-" term "-")+ term
graph ::= unit? { term* }
base ::= unit | string
unit ::= int
```

TODO: Ensure this is actually LL(1)! Probably need to massage some productions.

LISTING 1. The grammar for Welkin, shown in BNF notation (see Definition 4.3.5). The terminals `int` and `string` are defined in Definition 3.1.2 and Definition 4.2.4, respectively

4.4. **Proof of LL(1) Membership.**

We now prove that the Welkin language is unambiguous by showing it is LL(1), a rich class of grammars that can be efficiently parsed. For more details, please consult [Aho+06].

Definition 4.4.3. Let G be a grammar.

Definition 4.4.4. A grammar is LL(1) if, given two distinct productions α, β:

-
-
- If $\beta \overset{*}{\Rightarrow} \varepsilon$...

Theorem 4.4.5. *Welkin’s grammar is* LL(1)*. Hence, this grammar is unambiguous, i.e., every string accepted by the language has exactly one derivation.*

Proof. Consider the corresponding LL(1) table... □

5. Semantics

5.1. ASTs.

- Semantics on ASTs
  - 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 units

5.2. Representations.

Now we develop the formal framework to discuss information in terms of units, enabling a complete mechanization of Welkin's meta-theory. To keep this section self-contained, we explicitly provide all recursive definitions.

Definition 5.2.0. The **alphabet of units** is $\mathcal{A}_{\text{unit}} = u \mid \mathcal{A}_{\text{word}}$. A **unit ID** is combination of symbols u_w, where $w \in \text{word}$.

Definition 5.2.1. A **free parameter** is a parameter given an associated ID. No further restrictions are imposed.

We now define representations recursively, using unit IDs and free parameters as the base case.

Definition 5.2.2. Units are recursively defined:
- **Base case:** IDs and free parameters are units.
- **Recursive step:**
  - **Parts:**: if $u_1, .., u_n$ are finitely many units, then so is their combination $\{u_1, ..., u_n\}$. A combination defined without a provided ID is called an **anonymous unit**.
  - **Representations:** If u, w, v are units, so is $v \longrightarrow u$. We say v **represents** u. or conversely, u is **represented by** v.

Key equalities:
- $u.\{\} = u$. Acts as a sort of * operator from other languages.
  - To use one level up: $.u$
- $\left(u \xrightarrow{v} w\right) \in x \Leftrightarrow x(u) \xrightarrow{x(v)} x(w)$, where $x(u)$ is $x.u$ if $u \in x$ or u otherwise.

Example 5.2.3. Consider a house with a dog, a cat, and a person. We can represent the house as unit `house`, the dog as unit `dog`, the cat by unit `cat`, and the person by unit `person`. In our Welkin file, we add, `house { dog, cat, person}`. The `person` has an internal concept of `pet` and uses it to represent both the `dog` and `cat`, which we write as `person { animal --> dog, animal --> cat}`, under the scope of `house`.

Parts of units are denoted as $u.u'$. Scoping is included to provide namespaces. Moreover, parts enable **interpretations**. We write $u \overset{v}{\longrightarrow} u'$ in case $u, v, (u \longrightarrow v) \in u'$, so u represents v **via** u'. In this case, we say u' is a **context** to $u \longrightarrow v$. Note that unlabeled representations can have multiple contexts.

Example 5.2.4. Consider the recursive definition of a binary tree: either it is a null (leaf) node, or it contains two nodes, left and right. We can model this as follows:

- First, create units for each of the notions: `tree {null, left, right}`.
- Next, we write, `tree { null --> .tree, .tree --> left, .tree --> right, {.left, .right} --> .tree}`. Notice that we refer to the *namespace*, thereby enabling recursion. By our scoping rules, writing `tree` would be a *new* unit.
- To impose that the left subtree is *distinct* from the right one, we can use symbols.

An important idea in this example is that the abstraction could be defined *first*, or a concrete model could. For this reason, the choice of how entities are represented is flexible.

Definition 5.2.5. A unit u is **non-trivial** if it is non-empty and does not contain all relations. A unit u is **coherent relative to a context** u' if $u + u'$, the union of these units, is non-trivial.

Remark 5.2.6. This definition is a natural generalization of consistency in first-order logic. We will frequently rely on this result throughout the thesis.

Theorem 5.2.7. *A representation is preserves information modulo $\equiv$ iff the representation modulo $\equiv$ is coherent.*

Remark 5.2.8. This theorem enables truth management via specific contexts, specified as units. The task of finding core truths is then free, left open to flexibility accommodate for any truth management.

5.3. **Universal Systems.**

Inspired by [Mes12], we prove that scoping is strictly more expressive than without.

Lemma 5.3.9. *Representations with interpretations are undefinable in terms of unlabeled representations.*

TODO: TODO: define the generalization to Padoa's Method clearer.

Proof. It suffices to note that representing partial computable functions requires combinations. But every transformation under unlabeled representations does not preserve these conditions, hence, representations with interpretations are not definable. $\square$

Theorem 5.3.10. *Any computable function and its trace under a given string can be represented by units. (TODO: make more precise.)*

Proof. We prove there is a natural embedding from the lambda calculus into our metatheory, using two parts:

- $\lambda x.f$ is represented as $\{x \to f\}$.
- $f.g$ is represented as the combination of units, or $\{f, g\}$.

Thus, any λ-term can be represented in te meta-theory, completing the proof. $\square$

Note that there are multiple ways to prove Theorem 5.3.4, infinitely in fact. This motivates the following definition.

Definition 5.3.11. A universal representation system (URS) is a unit that can represent any representation.

Theorem 5.3.12. *A unit is a universal representation system if and only if it can represent any partial computable function. Moreover, any universal representation system can represent any universal representation system. In particular, representing itself is called **reflection**.*

TODO: Make this more precise and complete proof.

The term *universal* is specifically for expressing *representations* symbolically. The free parameter still needs to be included and is an additional feature on top of partial comptuable functions. However, the *management* of these symbols is done entirely with partial computable functions.

The next section discusses the issue of *managing* the infinitely many choices for URSs.

6. 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…

6.1. 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 6.1.0. *With respect to the class of all computable transformations that can be computably verified, there is no optimal formal system.*

6.2. **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.

7. 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!

8. 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

[Aho+06] Aho, Alfred V., Lam, Monica S., Sethi, Ravi, and Ullman, Jeffrey D., *Compilers: Principles, Techniques, and Tools (2nd Edition)*, Addison Wesley, 2006.

[Arc+06] Archambault, Éric, Vignola-Gagné, Étienne, Côté, Grégoire, Larivière, Vincent, and Gingras, Yves, Benchmarking scientific output in the social sciences and humanities: The limits of existing databases, Scientometrics **68** (2006) 329–342.

[Atk23] Atkin, Albert, Peirce's Theory of Signs, Spring2023 ed., Metaphysics Research Lab, Stanford University, 2023.

[AJ23] Auth, Gunnar, and Jokisch, Oliver, A systematic mapping study of standards and frameworks for information management in the digital era, Online Journal of Applied Knowledge Management **11** (2023) 1–13.

[Bra23] Bradley, F. H., The Principles of Logic, Mind **32** no. 127 (1923) 352–356.

[Bur09] Burgin, Mark, *Theory of Information*, World Scientific, 2009.

[CF09] Carette, Jacques, and Farmer, William M., A Review of Mathematical Knowledge Management, in *Intelligent Computer Mathematics* (Carette, Jacques, Dixon, Lucas, Coen, Claudio Sacerdoti, and Watt, Stephen M. (eds.), Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 233–246.

[Car+21] Carette, Jacques, Farmer, William M., Kohlhase, Michael, and Rabe, Florian, Big Math and the One-Brain Barrier: The Tetrapod Model of Mathematical Knowledge, The Mathematical Intelligencer **43** no. 1 (2021) 78–87.

[Car+21] Carroll, Stephanie Russo, Herczog, Edit, Hudson, Maui, Russell, Keith, and Stall, Shelley, Operationalizing the CARE and FAIR Principles for Indigenous Data Futures, Scientific Data **8** no. 1 (2021) 108.

[CF14] Casadevall, Arturo, and Fang, Ferric C., Specialized Science, Infection and Immunity **82** no. 4 (2014) 1355–1360.

[Dic25] Dictionary, Oxford English, welkin, n., Oxford University Press, 2025.

[Edw22] Edwards, John S., Where knowledge management and information management meet: Research directions, International Journal of Information Management **63** (2022) 102458.

[FG13] Fanelli, Daniele, and Glänzel, Wolfgang, Bibliometric Evidence for a Hierarchy of the Sciences, Plos One **8** no. 6 (2013) 1–11.

[Gui+25] Guillen-Aguinaga, Miriam, Aguinaga-Ontoso, Enrique, Guillen-Aguinaga, Laura, Guillen-Grima, Francisco, and Aguinaga-Ontoso, Ines, Data Quality in the Age of AI: A Review of Governance, Ethics, and the FAIR Principles, Data **10** no. 12 (2025).

[Har+20] Harrower, Natalie, Quinn, Mary, Tóth-Czifra, Erzsébet, and others, *Sustainable and FAIR Data Sharing in the Humanities: Recommendations of the ALLEA E-Humanities Working Group*, Berlin, 2020.

[Hed+19] Hedges, Mark, Stuart, David, Tzedopoulos, George, Bassett, Sheena, Garnett, Vicky, Giacomi, Roberta, and Sanesi, Maurizio, *Digital Humanities Foresight: The future impact of digital methods, technologies and infrastructures*, GOEDOC, Dokumenten-und Publikationsserver der Georg-August-Universität Göttingen, 2019.

[Jac+20] Jacobsen, Annika, Miranda Azevedo, Ricardo de, Juty, Nick, Batista, Dominique, Coles, Simon, Cornet, Ronald, Courtot, Mélanie, Crosas, Mercè, Dumontier, Michel, Evelo, Chris T., Goble, Carole, Guizzardi, Giancarlo, Hansen, Karsten Kryger, Hasnain, Ali, Hettne, Kristina, Heringa, Jaap, Hooft, Rob W.W., Imming, Melanie, Jeffery, Keith G., Kaliyaperumal, Rajaram, Kersloot, Martijn G., Kirkpatrick, Christine R., Kuhn, Tobias, Labastida, Ignasi, Magagna, Barbara, McQuilton, Peter, Meyers, Natalie, Montesanti, Annalisa, Reisen, Mirjam van, Rocca-Serra, Philippe, Pergl, Robert, Sansone, Susanna-Assunta, Silva Santos, Luiz Olavo Bonino da, Schneider, Juliane, Strawn, George, Thompson, Mark, Waagmeester, Andra, Weigel, Tobias, Wilkinson, Mark D., Willighagen, Egon L., Wittenburg, Peter, Roos, Marco, Mons, Barend, and Schultes, Erik, FAIR Principles: Interpretations and Implementation Considerations, Data Intelligence **2** nos. 1–2 (2020) 10–29.

[KR16] Kohlhase, Michael, and Rabe, Florian, QED Reloaded: Towards a Pluralistic Formal Library of Mathematical Knowledge, Journal of Formalized Reasoning **9** (2016) 201–234.

[The20] The mathlib Community, The lean mathematical library, in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Association for Computing Machinery, New Orleans, LA, USA, pp. 367–381.

[Mes12] Meseguer, José, Twenty years of rewriting logic, The Journal of Logic and Algebraic Programming **81** no. 7 (2012) 721–781.

[Mik23] Mikkilineni, Rao, Mark Burgin's Legacy: The General Theory of Information, the Digital Genome, and the Future of Machine Intelligence, Philosophies **8** no. 6 (2023).

[Wil+16] Wilkinson, Mark D., Dumontier, Michel, Aalbersberg, IJsbrand Jan, Appleton, Gabrielle, Axton, Myles, Baak, Arie, Blomberg, Niklas, Boiten, Jan-Willem, Silva Santos, Luiz Olavo Bonino da, Bourne, Philip E., Bouwman, Jildau, Brookes, Anthony J., Clark, Tim, Crosas, Mercè, Dillo, Ingrid, Dumon, Olivier, Edmunds, Scott C., Evelo, Chris T. A., Finkers, Richard, González-Beltrán, Alejandra N., Gray, Alasdair J. G., Groth, Paul, Goble, Carole A., Grethe, Jeffrey S., Heringa, Jaap, Hoen, Peter A. C. 't, Hooft, Rob W. W., Kuhn, Tobias, Kok, Ruben G., Kok, Joost N., Lusher, Scott J., Martone, Maryann E., Mons, Albert, Packer, Abel Laerte, Persson, Bengt, Rocca-Serra, Philippe, Roos, Marco, Schaik, Rene C. van, Sansone, Susanna-Assunta, Schultes, Erik Anthony, Sengstag, Thierry, Slater, Ted, Strawn, George O., Swertz, Morris A., Thompson, Mark, Lei, Johan van der, Mulligen, Erik M. van, Velterop, Jan, Waagmeester, Andra, Wittenburg, Peter, Wolstencroft, Katy, Zhao, Jun, and Mons, Barend, The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data **3** (2016).

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