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	**Syntax**	Provides the grammar and proof that it is unambiguous.
Section 4	**Semantics**	Explains how ASTs are validated and processed. Develops representations and coherency, and connects these to a working definition of information.
Section 5	**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 6	**Bootstrap**	Bootstraps the language.
Section 7	**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. Syntax

We keep this section self-contained with explicit alphabets and recursive definitions. For general notation, we write $a_0, \ldots, a_n$ for a finite list of items, and use $a ::= a_1 \mid \ldots \mid a_n$ to denote a definition of a in terms of $a_1, \ldots, a_n$. that for verification purposes, we will incorporate fixed bounds into Section 6. Moreover, we will postpone discussions on the rationale for the simple syntax until Section 4.

3.1. **Words.**

Welkin's main encoding uses binary words, but add notation for decimal and hexadecimal.

```
bit ::= 0 | 1
digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
nibble ::= A | B | C | D | E | F
```

Listing 1. Binary, decimal, and hexadecimal digits.

A word is a sequence of digits, see Listing 2. We leave concatenation `;` as an undefined notion. We set concatenation to be right-associative, i.e., $(w.w') + w'' = w.(w'.w'')$, and abbreviate $w.w'$ as ww'. For conversions, see Definition 4.2.3.

```
word --> binary | decimal | hex
binary --> bit | binary.bit
decimal --> digit | decimal.digit
hex --> nibble | hex.nibble
```

Listing 2. Definition of words.

3.2. **Terminals.**

Welkin uses ASCII as its base encoding. The term ASCII is slightly ambiguous, as there are subtly distinct dialects, so we formally define US-ASCII as a standard dialect.[1]

Definition 3.2.0. US-ASCII consists of 256 symbols, listed in Table 3.

To represent general encodings, there is a binary format supported for strings, see Listing 4.

[1]Note that this table *itself* is a representation, which represents glyphs with binary words. The use of these kinds of representations occur frequently in Welkin, see Section 6.

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.

There are several important character classes.

```
PRINTABLE  ::= [0x20-0x7E]
WHITESPACE ::= [0x09, 0x0A, 0x0D, 0x20]
DELIMITER ::= [0x7B, 0x7D, 0x2C, 0x2D, 0x2A, 0x3C, 0x3E, 0x22, 0x27, 0x5C,
0x7D]
```

LISTING 3. Important character classes.

```
STRING ::= SQ_STRING | DQ_STRING

SQ_CHAR ::= PRINTABLE \ {'}
DQ_CHAR ::= PRINTABLE \ {"}
```

LISTING 4. Strings.

```
IMPORT ::= "@" ID
ID :: ID_CHAR+
ID_CHAR ::= PRINTABLE / (DELIMITERS + WHITESPACE)
```

LISTING 5. IDs.

3.3. **Grammar.**

Welkin's grammar is displayed in Listing 6, inspired by a minimal, C-style syntax.

```
start ::= (term ",")* term
term ::= arc | graph | path
arc ::= (term "-" term "->)+ term
      | (term "<-" term "-")+ term
      | (term "-" term "-")+ term
graph ::= path? { term* }
path ::= unit | "."."*" | "."+
dots ::= "." dots*
unit ::= IMPORT | ID | STRING
```

LISTING 6. The grammar for Welkin. The terminals `id` and `string` are defined in Listing 2 and Listing 4, respectively

Note that we are interested in *transducers*, or having the parser generate a parse tree from a given string. We define parse trees recursively in Listing 7.

```
node ::= leaf | node [node1, node2, ..., noden]
leaf ::= t in T | epsilon
```

LISTING 7. Representation of a parse tree as a list.

3.4. **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].

Moreover, we define the top of a word in Listing 8.

```
top(word) ::= nil => nil | bit.word => bit
```

LISTING 8. Definition of the top of a word.

Definition 3.4.1. ([RS70]). A grammar is LL(1) iff the following holds: for any terminals w_1, w_2 and nonterminal A, there is at most one rule r such that for some w_2, w_3,

- $S \Rightarrow \mathrm{top}(w_1)Aw_3$
- $A \Rightarrow w_2(p)$
- $\mathrm{top}(w_2w_3) = w$

Theorem 3.4.2. *Welkin's language is accepted by some* LL(1) *grammar. Hence, Welkin's syntax is unambiguous, i.e., every string accepted by the language has exactly one derivation.*

Proof. We left-factor common prefixes, resulting in Listing 9. For the refactor step by step, see Table 4. We can readily verify that there are no shared prefixes for a single production, see Table 5.

```
start ::= (term ",")* term
term ::= arc | graph | path
arc ::= (term "-" term "->)+ term
      | (term "<-" term "-")+ term
      | (term "-" term "-")+ term
graph ::= path? { term* }
path ::= unit | "."."*" | "."+
dots ::= "." dots*
unit ::= IMPORT | ID | STRING
```

LISTING 9. Transformed LL(1) grammar for Welkin.

Original	**Transform**	**LL(1)**

TABLE 4. Step by step refactor for the

TABLE 5. LL(1) Table for Listing 9

□

4. SEMANTICS

This section describes how ASTs are processed and validated. We postpone information organization to Section 5.

4.1. **Rationale.**

We justify why the language is focused on representations. First, to mechanize the information language, we allow only total computable functions, with computability being a well established notion. Second, to enable clarity in concepts, we need to resolve the Symbol Grounding Problem, so as to avoid treating all symbols as being "empty", as discussed in [Liu25]. We must therefore include a notion of representation, which, in particular, can represent partial computable functions. Finally, we claim that expressing *any computable representation* is sufficient for a universally expressible information system. Attempting to provide a self-contained definition of the notion "any" is problematic, as shown from the introdution. We instead define "any" with the *least* restrictions possible, which means, by the first point, ensuring that a given provided input is accepted by *some* computable function. It is important that Welkin includes *every* computable function in this definition, which we prove in Theorem 4.4.4.

4.2. **ASTs.**

Given the rationale, we explain how the Abstract Syntax Tree (AST) is processed for the syntax. The AST provides an intermediate step before the final data structure.

Hexadecimal	Decimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

TABLE 6. Conversions of digits between different bases.

Note that for words, we add a conversion from decimal and hexadecimal into binary via Table 6. We provide the explicit recursive definiton based on this table in Listing 10, where `a <--> b` means that `a` is converted into `b` and vice versa. This is a restriction on the notion of representations that will be addressed in Section 6.

LISTING 10. Recursive definition for converting words between bases.

```
"0".word <--> word
"0b0".word <--> "0b".word
"0x0".word <--> "0x".word
```

LISTING 11. Conversions with leading zeros.

Definition 4.2.0. The AST is recursively defined from the parse tree as follows:

- **Arc:** Converts a chain into a list of tuples of the form (sign, context, referant). Renders each edge as a left and right arrow.
- **Graph:** The terms are collected into two parts: a list of parts and a list of arcs.
- **Path:**
  - The number of dots is counted for the relative paths.
  - Star imports are denoted by a special node All.
  - A path is converted into a list of its contents,

  which are pairs containing the relative path number and either Unit or All.

The terms in the top-level are put into a Graph node containing a unique, user given ID.

Definition 4.2.1. An AST is **valid** if the relative imports does not exceed the number of available parents.

Remark 4.2.2. An earlier revision of this thesis forbid repetitions of arcs and units. However, this restriction was removed to provide greater flexibility. This will be tracked, see ?.

4.3. **Faithful Representations and Truth Management.**

Based on Section 4.1, a crucial question is to answer *how* representations can be used in the language. A representation at least contains two components: a *sign* that represents a *referant*. However, this is not sufficient to express any computable function, because we do not have *conditional* representations. A key insight in this thesis is showing that expressing conditions is equivalent to having *contexts*, which we incorporate into our mechanism for namespaces and generalizes Burgin's notion of infological systems [Bur09]. This proves an informal claim made in Meseguer [Mes12], which claims that rewriting logics without conditional rules are "strictly less" expressive than those with conditions, see Theorem 4.3.3.

We define a *unit* as an extendible component in a representation that can be broken down, build new units, or act on other units. Computationally, we can treat units as IDs to partial computable functions, but we permit *implicit bindings* to non-symbolic things (a term made vague for flexibility). From there, we practically formalize information being *contained* in a unit, enabling change in a context through checking for some *non*-fixed point. This connects to Burgin's analogy of information as energy, as well as Bateson's famous quote that "information is a difference that makes a difference" [Bat00].

Definition 4.3.3. A **unit** is defined from the AST as follows... The **combination** of units u, u', denoted by $u + u'$ is defined to be the pairwise union of components across. Note that is different from the **disjoint union**, in which a new top level node is made with children u and u'.

We set $\left(u \xrightarrow{v} w\right) \in x \Leftrightarrow x(u) \xrightarrow{x(v)} x(w)$, where $x(s)$ is the local extension of s in x. We interpret $u \xrightarrow{c} v$ as: the **sign** u represents **referant** v in **context** c.

Definition 4.3.4. A unit u is **non-trivial** if it is non-empty and has a non-complete representation graph. A unit u is **coherent relative to a context** u' if $u + u'$, the union of these units, is non-trivial.

Remark 4.3.5. This definition is a natural generalization of consistency in first-order logic. We will frequently rely on this result throughout the thesis.

Definition 4.3.6. Let u, v be units. Then u **contains information** v if for some $s \in v$, $u[s] \neq s$.

Our notion of information helps with one key issue: the general undefinability of non-trivial classes of partial computable functions in formal system. This connects with the absence of a universal *single* formal system that can prove any claim about, e.g., Peano Arithmetic.

Example 4.3.7. Trees.

A key technique in managing information and truth through contexts is through the following theorem. FIXME: this is currently a stub! Need to create the **correct** condition. Use this as a starting point:

Theorem 4.3.8. *A unit u contains information about v iff $u + v$ is coherent.*

Theorem 4.3.9. *Representations with contexts cannot be expressed with those without.*

Proof. The largest class expressible with unconditional representations are context-free grammars, because... Thus, not all partial computable functions are included, completing the proof. □

4.4. **Universal Systems.**

Theorem 4.4.10.

As a consequence, we immediately obtain the following corollary.

Corollary 4.4.11. *Every truth management system, accepted by some computable function, is definable as a unit.*

Note that there are multiple ways to prove Theorem 4.4.4, infinitely in fact. This motivates the following definition.

Definition 4.4.12. A universal representation system (URS) is a unit that can represent any representation.

Theorem 4.4.13. *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**.*

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.

5. 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…

5.1. **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 5.1.0. *With respect to the class of all computable transformations that can be computably verified, there is no optimal formal system.*

5.2. **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.

6. Bootstrap

This section proves that there is a file, which we call `weklin.welkin`, that contains enough information to *represent* Welkin. We do not bootstrap proofs in this thesis, but that could easily be a future extension.

6.1. **Self-Contained Standard.**

This section is self-contained and defines *everything* necessary about Welkin. The complete bootstrap is in appendix ?.

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

[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.

[Bat00] Bateson, Gregory, *Steps to an ecology of mind*, University of Chicago Press ed., University of Chicago Press, Chicago, 2000.

[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.

[Liu25] Liu, Zhangchi, An Algorithmic Information-Theoretic Perspective on the Symbol Grounding Problem, arXiv e-prints (2025) arXiv:2510.05153.

[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).

[RS70] Rosenkrantz, D.J., and Stearns, R.E., Properties of deterministic top-down grammars, Information and Control **17** no. 3 (1970) 226–256.

[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