Data Standards for Artificial Life Software

Alexander Lalejini[1,2,3,*], Emily Dolson[1,2,3,*], Clifford Bohm[1,2,4,*], Austin J. Ferguson[1,2,3], David P. Parsons[5], Penelope Faulkner Rainford[6], Paul Richmond[7], Charles Ofria[1,2,3,*]

[*]Organizing author
[1]BEACON Center for the Study of Evolution in Action
[2]Ecology, Evolutionary Biology, and Behavior Program, Michigan State University
[3]Department of Computer Science, Michigan State University
[4]Department of Integrative Biology, Michigan State University
[5]Inria Beagle Team, F-69603, France
[6]Department of Mathematical Sciences, Durham University
[7]Department of Computer Science, University of Sheffield
lalejini@msu.edu

Abstract

As the field of Artificial Life advances and grows, we find ourselves in the midst of an increasingly complex ecosystem of software systems. Each system is developed to address particular research objectives, all unified under the common goal of understanding life. Such an ambitious endeavor begets a variety of algorithmic challenges. Many projects have solved some of these problems for individual systems, but these solutions are rarely portable and often must be re-engineered across systems. Here, we propose a community-driven process of developing standards for representing commonly used types of data across our field. These standards will improve software re-use across research groups and allow for easier comparisons of results generated with different artificial life systems. We began the process of developing data standards with two discussion-driven workshops (one at the 2018 Conference for Artificial Life and the other at the 2018 Congress for the BEACON Center for the Study of Evolution in Action). At each of these workshops, we discussed the vision for Artificial Life data standards, proposed and refined a standard for phylogeny (ancestry tree) data, and solicited feedback from attendees. In addition to proposing a general vision and framework for Artificial Life data standards, we release and discuss version 1.0.0 of the standards. This release includes the phylogeny data standard developed at these workshops and several software resources under development to support our proposed phylogeny standards framework.

Introduction

Artificial Life (ALife) research is becoming more complex as the field advances and as computational power increases. Further, more recent initiatives have broadened the scope of the field to intersect topics such as society and education, attracting new and interesting perspectives to the community. We find ourselves in the midst of an increasingly complex ecosystem of ALife software systems (Taylor et al., 2016), including research platforms, metrics, data visualizations, *et cetera*. Each system is developed to address particular research objectives, all unified under the common goal of understanding life; such a monumental goal begets a number of algorithmic challenges (*e.g.*, tracking a single gene through a genetic lineage, measuring the open-endedness of a system, identifying and characterizing complex interactions among individuals in a population). Many projects have solved some of these problems in individual systems, but these solutions are rarely portable and often must be re-engineered across systems.

Many other communities have developed and leveraged data standards to dramatically improve their software ecosystems. Data standards are specifications for organizing, annotating, and recording commonly-collected information. That is, what specific values should we keep, what descriptors (properties) should we use to specify them, and in what format should they be stored?

We propose a community-driven process of developing such standards for Artificial Life in an effort to improve software re-use and allow for easier comparisons of data generated with different artificial life systems. Standards allow tools to be developed that can immediately be applied to data produced by unrelated systems, eliminating the need for these tools to be re-written by each research group. In addition to saving time, expanding the user base for tools increases their reliability by making it harder for bugs to go undetected. Moreover, creating a collaborative software ecosystem will facilitate communication and cooperation among research groups by making it easier to compare results across different systems using the same analysis tools. Further, standards increase the incentive to develop tools that solve elusive community-wide challenges, as you will be able to immediately apply them to a broad cross-section of available systems and data; likewise, many fellow researchers will be able to make easy use of your tools.

We began the process of developing data standards with two discussion-driven workshops at the 2018 Conference for Artificial Life and the 2018 Congress for the BEACON Center for the Study of Evolution in Action. At both work-

shops, we discussed the vision for ALife data standards, proposed and refined a standard for phylogenies (that is, a standard for describing parent-offspring relationships over time), and solicited feedback from attendees (Lalejini and Dolson, 2019); in conjunction, we developed software tools to leverage these proposed standards. This paper is a continuation of these efforts. Here, we provide examples of how data standardization has benefited other scientific communities. We summarize our vision for artificial life data standards, and propose a framework for ALife data standards. By way of example, we present the phylogeny standard discussed in both 2018 workshops; additionally, we identify several existing software resources under development to support our proposed phylogeny standards framework: developer utilities, data converters, and end-user tools. We conclude with a discussion of future directions, including possible concepts for future standardization.

The Benefits of Data Standardization: Examples From Other Communities

As scientific communities extend their reach, data standards provide a mechanism to unify software development and provide a better user experience. Data standards afford developers a reduced barrier to entry, the ability to more easily communicate across disciplines, and a broader impact from their software efforts. As such, users experience a more unified software ecosystem where they can use the same analysis and visualization tools across research platforms. Our vision for Artificial Life data standards is inspired by these other successful efforts.

Data standards have been successfully adopted throughout history. For example, the metric system revolutionized how weights and other measures are used throughout science, and failures to keep to this standard have proved catastrophic (Board, 1999). The ASCII standard shaped how modern computers manage text, allowing developers to write versatile tools to manipulate human-readable files. In modern biology, both computational neuroscience (Gleeson et al., 2010; Richmond et al., 2014) and systems biology (Hucka et al., 2003) have adopted successful data standards. In both cases, these standards were driven by an open community approach, resulting in improved model exchange and design as well as the development of compliant simulators. Furthermore, digital access to global data is integral to biodiversity research. The Darwin Core project data standards (Wieczorek et al., 2012) define relevant properties for a range of scientific entities (*e.g.*, taxa, occurrences, fossil specimens); this standardization has eased communication of and collaboration using biodiversity data, allowing the community to homogenize biodiversity record structure across multiple repositories (Parr et al., 2012). The bioinformatics community is moving toward widely-adopted data standards (Zhang et al., 2011), ushering the development of broadly used (Wren, 2016) databases and software tools.

Highly relevant to the field of Artificial Life, the Robot Operating System (ROS) is a popular, open-source software development framework that defines communication and data standards. The ROS standards have facilitated massive community software development, sharing, and reuse (Quigley et al., 2009). By defining common standards for software, ROS unifies disparate sub-communities (ranging from academic researchers to industrial engineers to hobbyists), creating the opportunity for robotics collaborations among people who would have never even communicated otherwise. In 2012, the ROS community began organizing annual ROScon events, an international conference where ROS software developers meet and present recent software applications, ideas, and tools. All of this community buy-in and support for software developers eases onboarding for new researchers and lowers the barrier to making meaningful software contributions for the community.

Our Vision for Artificial Life Data Standards

Data standards specify how data are described (ontologies) and recorded (formats). Because the types of data used in Artificial Life research are many and varied (from experiment-to-experiment and system-to-system), any useful set of standards for our community will need: 1) a minimalist and inclusive core shared by all, 2) a flexible mechanism for extensions to encompass the idiosyncrasies of individual systems, and 3) enough descriptive power to allow for tools that will be broadly useful.

Because ALife systems and experiments are diverse, we must ensure that standards for *describing* different types of data are flexible. We envision that each data standard should minimize the number of properties required to specify the concept of interest; we should avoid incorporating extraneous or restrictive assumptions into the standards, ensuring the core of each standard remains inclusive. For example, to meaningfully describe a lineage, we require, at minimum, information about parent-offspring relationships. While broadly applicable, such a data standard would sacrifice utility if it disallowed extra information of potential interest such as organism characteristics or mutational changes that occurred along the lineage. Thus, in addition to a set of mandatory properties, each data type will also standardize a set of optional (or "conventional") properties. Standard-compliant data are not required to report optional information; however, *if* standardized optional properties are included, they must use the specified labels and formatting. Each standard-compliant software tool must document which optional properties it accepts and/or requires, and be able to ignore those that it does not use.

The diversity of systems and experimental settings not only requires flexibility on *what* data may be recorded but also on *how* it should be recorded (*i.e.*, the underlying file formats). Depending on the data itself and on what one wants to do with it, the choice of file format could range

from a verbose format such as JSON or XML to a compressed binary format. Our vision is for the standards to support multiple, interchangeable file formats (*e.g.*, XML, JSON, CSV, binary, *etc.*) to ensure that data recording is maximally flexible. Any of these formats could be used to store data and provide input to standard-compliant software tools; this flexibility, however, demands that we provide detailed guidelines for conversions between storage formats.

Proposed Standards Framework

Our proposed standards framework specifies: 1) the terminology for referring to specific concepts (ontology), 2) a structure for describing data, 3) rules for formatting data, and 4) the process for creating and modifying the standards. Note that our proposed framework does not constrain methods for analyzing or working with data. However, agreeing on standardized ways of formatting and describing data will ease the development of analysis tools and visualizations.

Ontology

A critical component of any data standard is a set of agreed-upon terminology for describing data, with clear rules about how information fits together. In information science, such a framework is called an *ontology* (Smith and Welty, 2001). For consistency, we will use the same terms that are used in other ontology development research.

Each individual standard in our framework specifies a way of describing and storing a particular instance of a *concept* (*e.g.*, the "phylogeny standard" describes how to store the concept of a phylogeny). Each instance of a concept can be described with a single, arbitrarily large data table where rows are *entities* (*e.g.*, individual taxa in a phylogeny) and columns specify *properties* (*e.g.*, ancestor IDs or trait values) of each entity; further, an individual data table may contain entities of only a single type or category.

Describing Data

Each standardized data type (concept) has three categories of properties (*i.e.*, data fields or attributes): *required* properties, *conventional* (or optional) properties, and *extra* (or additional) properties. To qualify as standard-compliant, a data file must abide by *all* required properties. Required properties are what the community determines to be the minimal set of properties needed to meaningfully specify the concept of interest. Properties should only be given required status if they are fundamental to the concept being represented, such as parent-offspring relationships in a phylogeny.

In addition to required properties, each concept will standardize a set of conventional properties. Conventional properties are used to describe pieces of data that are often important, but are not fundamental to the concept being recorded. As such, standard-compliant systems are not required to output conventional properties (and, indeed, these properties may not even be meaningful in all systems or setups). If you do choose to output conventional properties and label them according to the standard, these properties may be leveraged by standard-compliant tools. For example, mutation counts are not required to record a phylogeny, but if they are included, analysis tools can produce more informative visualizations indicating amount of change over time.

Extra properties include any data not otherwise specified; these are system-specific or experiment-specific properties that further describe the concept. Allowing for arbitrary extra properties ensures the standard is inclusive and easy to use. Software tools should document any extra properties they can make use of and their meaning. In the event that an extra property is used by multiple software systems, it may be appropriate to formalize it as a conventional property. Additionally, many software tools may be able to use arbitrary properties by name. For example, a user may be able to choose an any property they want to color-code a phylogeny.

Property names must be consistent across file formats. As of version 1.0.0, all property names are in snake case: fully lowercase (*e.g.*, 'id' instead of 'ID' or 'Id') and underscore-separated as appropriate (*e.g.*, 'ancestor_list'). When deciding on required or conventional property names, we will err on the side of being descriptive to ensure that files remain intuitive. We encourage extra properties to be similarly descriptive to simplify data sharing, limit name collisions, and facilitate future conversion to conventional property status.

Naming Modifiers Workshop participants suggested we specify conventions for naming common *types* of properties (*e.g.*, lists, averages, variances, *etc.*). For example, the property for identifying the set of offspring produced by a particular organism might be called 'offspring_list' instead of 'offspring' to indicate that the property refers to a list. We envision the set of these conventions (*i.e.*, naming modifiers) to grow as new tools are developed and as the standards grow to encompass more concepts. Table 1 provides the set of naming modifiers in version 1.0.0 of the standard. Naming modifiers should be applied to the end of the property name, connected via an underscore (*e.g.*, 'offspring_list'). These conventions allow tools to be more flexible when loading and processing standardized data by inferring data types for properties and identifying property relationships from a common prefix. For example, if a tool sees both fitness_ave and fitness_std, it may reasonably assume that these refer to the average and standard deviation of the same distribution.

Reserved and Default Values For certain properties, it is valuable to reserve values to have special meanings. For example, what value should ancestor_list use to indicate that an organism in a phylogeny was created randomly and therefore has no ancestors? This parameter could be left empty, but how would we be able to differentiate this organism from one that migrated from another population or from an or-

Table 1: Proposed conventional property name suffixes.

Suffix	Description of Property
_id	A unique identifier, often numerical.
_name	A string label identifier.
_count	A whole number count.
_total	A cumulative result of counts over time.
_list	A list of values.
_sum	The summed total of a list of values.
_time	A numerical measure of time.
_rate _prob	A rate or probability.
_ave _med _min _max _var _std _skew _kurt	Measurements of an observed or calculated distribution (average, median, minimum, maximum, variance, standard deviation, skew, and kurtosis).

ganism that was loaded from a previous experiment? The standard can specify a set of conventional reserved values, giving standard-compliant tools a way to recognize and differentiate these special cases.

Just as it is useful to reserve data values, it may also be useful for the standard to specify reasonable default values for certain conventional (non-required) properties. Accepted and well-documented default values allow tools to make predictable assumptions in cases where conventional properties are missing.

Formatting Data

Our proposed standards framework supports multiple formats for recording and storing data. Every standard-compliant tool will be required to support at least one standard-compliant file types as input; we will curate tools capable of converting data between supported file formats. As of version 1.0.0, the standards support JavaScript Object Notation (JSON) and Comma Separated Values (CSV) formats. As demand builds for additional file formats, the community can develop rules for representing standard data in these new formats (along with converters to and from already-supported formats).

Process for Amending Standards

In the long run, we will model the ALife data standards on other open source projects. As the number of standard-compliant tools increases, so too will the number of people who are invested in their maintenance and improvement. Thus, it is important to establish a process for amending the standards. Our initial set of standards are housed in a repository on GitHub (Lalejini et al., 2019). Anyone can suggest an update to the standards by submitting a pull request or issue. Proposed changes will be reviewed and discussed by interested community members to ensure that 1) they are backwards-compatible (unless there is a compelling reason for a breaking change), 2) they do not replicate or clash with existing standards, and 3) they are well specified, inclusive, and flexible. Following this discussion and a positive consensus, the changes will be merged in. This process follows the successful precedent set by other open source standards (Darwin Core task group, 2014). We anticipate that additions to the standards will be driven by tool developers adopting conventions for how to name specific types of data.

Changes to the standards will be tracked using semantic versioning, a system of assigning version numbers that conveys information about how similar successive versions are. Versions are identified with a sequence of three numbers (*e.g.*, as of this paper, we are on version 1.0.0), which indicate the "major version", "minor version", and "patch" respectively. Typically, a change in the major version denotes a break in backwards compatibility, a change in the minor version denotes the addition of new backwards-compatible features, and a change in the patch denotes minor bug fixes.

Proposed Phylogeny Standard

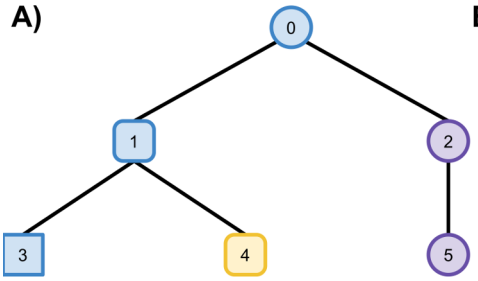
The problem of how to represent a phylogeny in a standardized way has long been important to biology, even without the wealth of data we have access to in ALife. Biologists have developed a few standard representations for phylogenetic trees (Cranston et al., 2014). Some popular formats include Newick (Cardona et al., 2008), Nexus (Maddison et al., 1997), NHX (Zmasek and Eddy, 2001), PhyloXML (Han and Zmasek, 2009) and RecPhyloXML (Duchemin et al., 2018). Newick format is a simple nested format for representing the hierarchy of a tree (a tree with three nodes might be represented as ((A,B),C), for example). Nexus and NHX formats build upon Newick with additional information (*e.g.*, the inclusion of a genetic sequence alignment among the represented taxa). PhyloXML and RecPhyloXML support the inclusion of additional supplemental data, but still use a nested format.

Why not use one of these existing standards for phylogenies in artificial life? These biological phylogeny standards were designed to work with species phylogenies inferred from extant taxa (and fossil data). As such, these standards are not designed to represent phylogenies with finer taxonomic scale. In artificial life, it is often reasonable to examine the complete phylogeny of every individual that ever existed. Attempting to do so presents two problems for phylogeny standards used in biology: 1) representing any phylogeny where taxa have multiple parents is impossible in these standards, and 2) complete phylogenies can be so big that it is necessary to split them across multiple files – nested formats (as used in biology) do not support such splitting.

During the workshops at ALife 2018 and the 2018 BEACON Congress (Lalejini and Dolson, 2019), we created a proposed standard for representing phylogenies. After continued discussion, we believe that it is now ready to be adopted.

Phylogenies depict parent-offspring relationships over the course of evolution. Phylogenies can be constructed for any taxonomic unit of organization (*e.g.*, individuals, genotypes, species, *etc.*); thus, we use the term "taxon" to refer gener-

A)



B)

id	ancestor_list	origin_time	destruction_time	shape	corners	color
0	[none]	0	1	circle	rounded	blue
1	[0]	1	2	square	rounded	blue
2	[0]	1	2	circle	rounded	purple
3	[1]	2	none	square	sharp	blue
4	[1]	2	none	square	rounded	yellow
5	[2]	2	none	circle	rounded	purple

Figure 1: (A) A simple phylogenetic tree where each entity's id is given inside of its 'colored-shape' phenotype. (B) A corresponding standard-compliant data table. The data includes required properties (id and ancestor_list), two optional properties (origin_time and destruction_time), and three extra properties that were used by the visualization (shape, corners, and color).

ally to an entity in a phylogeny. Each taxon in the file must have existed at some point, and each relation from one entity to another defines an ancestor-descendant relationship between the two entities (taxa).

The phylogeny standard has two mandatory properties: **id** and **ancestor_list**. The **id** property provides a unique identifier corresponding to that taxon. The **ancestor_list** property contains a list of ids corresponding to ancestors of the taxon. These are not required to be the direct parents of the taxon, but they will usually be treated as the closest ancestors in the phylogeny. All ids in the **ancestor_list** must correspond to taxa in the file. In cases where a taxon has no ancestors in the file, non-numeric string values can be used to specify that taxon's origin.

Version 1.0 of the phylogeny standard has two optional properties: **origin_time** and **destruction_time**, which specify the time that the taxon came into and out of existence, respectively. Setting these properties to strings also allows for special values, such as a keyword for **destruction_time** to indicate that a taxon is still alive. Figure 1 gives an example of a phylogeny and its associated standard-compliant description.

Current Software Support

In addition to housing the Artificial Life data standards specifications in a GitHub repository, we plan to maintain a community-curated list of standard-compliant software tools on GitHub (`https://github.com/alife-data-standards/alife-data-tools`; Bohm et al. 2019). This list of software resources is modeled after other efforts to use GitHub as a platform for selecting, evaluating, and organizing public resources for preservation and future use (Wu et al., 2017). Anyone can suggest an update to the list of software resources by submitting an issue or a pull request; for example, a student who has developed a useful visualization tool and written an explanatory blog post would be able to submit an issue to have a link to their visualization and blog post added to the list of software resources.

While concentrating developer effort on a single set of tools will make those tools more reliable, it also increases the harm that any individual bug in the software can do. To minimize this risk, we advocate the use of software development best practices. All repositories maintained by the Artificial Life Data Standards Organization use continuous integration to ensure that all code is automatically tested when a change is made. Test coverage is measured to facilitate these test suites in becoming comprehensive. Lastly, static analysis is automatically performed to identify error-prone code. Software on the resource list will be classified by reliability based on how well it follows these best practices.

Thus far, we have developed (and are continuing to develop) a variety of software resources to support our proposed phylogeny standard. These resources fall under three broad categories (that are not necessarily mutually exclusive): developer utilities, data converters (to and from the standard), and end-user tools. In addition to being useful on their own, we intend for these resources to serve as templates for developing new standard-compliant software tools. As we develop more software support, we will document them on our list of standard-compliant software tools on GitHub (Bohm et al., 2019). The rest of this section discusses each of the software tools already developed that work with the ALife data standards.

Developer Utilities

Developer utilities include software packages and libraries that can be incorporated into new tools. Thus far, we have begun development on a Python package for working with standardized phylogeny data. We plan to produce a similar set of tools in C++. As additional ALife standards are released, tools to assist developers using that standard will encourage broader adoption.

ALife Standards Development Python Package The ALife data standards Python package includes functions for loading a standard-formatted phylogeny file as a NetworkX (Hagberg et al., 2008) directed graph object. Python is a popular language for many tasks including data manipulation and analysis; further, many efforts have been made to build interfaces from Python to other languages (*e.g.*, Allaire et al. 2018; Guelton et al. 2015). These benefits, in

addition to its ease of use, make Python an ideal language for developing this initial package of software utilities.

NetworkX is a popular Python package for creating and manipulating graphs. By representing phylogenies as NetworkX graph objects, we can apply existing graph algorithms and visualizations to our phylogenies. Additionally, our Python package contains utilities for saving, manipulating, and analyzing phylogenies and lineages. See `https://github.com/alife-data-standards/alife-std-dev-python` for a more detailed description of this package's current functionality.

Data Converters

Data conversion tools translate data files between formats. Data converters may allow one to use standard-compliant analysis or visualization tools on data produced by non-standard-compliant systems and vice-versa. Data converters may also be developed to translate between different encodings of ALife standard data (*e.g.*, from JSON to CSV, each of which having their pros and cons). We envision data converters serving as the bridge between otherwise incompatible software tools and systems. Thus far, we have developed three data conversion utilities:

Avida to Standard Phylogeny In the Avida Digital Evolution Platform (Ofria and Wilke, 2004), self-replicating computer programs compete, mutate, and evolve. Avida has been used to study a wide range of evolutionary dynamics (*e.g.*, Goldsby et al. 2012; Zaman et al. 2014; Dolson et al. 2016). By default, Avida outputs population files at regular intervals during an experiment. Each population file contains information about the genotypes present in the current population as well as the full ancestral lineages for each extant genotype. Our Avida to standard phylogeny converter takes a single Avida population file as input and converts it into the standard phylogeny format (either as CSV or JSON). This converter and more detailed usage information can be found on GitHub at `https://github.com/alife-data-standards/converters-avida`.

MABE to Standard Phylogeny The Modular Agent-based Evolver (MABE) is a software framework developed to support research in digital evolution and artificial life (Bohm and Hintze, 2017). MABE allows researchers to construct experiments by combining different types *modules*: genomes, brains, environments, and selection methods. These modules can be drawn from an ever-growing collection or be developed by the user as necessary.

MABE outputs ancestry information in a series of population snapshot files (either full snapshots or pruned snapshots without reproductively unsuccessful individuals). The MABE to standard phylogeny converter takes these MABE snapshot files and optionally a list of column names that should be included in the standard phylogeny output. This converter and more detailed usage information can be found on GitHub at `https://github.com/alife-data-standards/converters-mabe`.

Standard Phylogeny to VINE The Visual Inspector for NeuroEvolution (VINE) (Wang et al., 2018) is under active development by UBER Labs in conjunction with their Deep Neuroevolution project (Such et al., 2017). VINE allows users to visualize how an evolving population moves through trait space over time. The standard phylogeny to VINE converter allows users to identify which properties of their phylogeny data should be translated into the VINE input format and creates the VINE-compliant input files (in the appropriate directory structure). For more information about VINE, see (Wang et al., 2018). More detailed usage information for our standard phylogeny to VINE converter can be found on GitHub at `https://github.com/alife-data-standards/converters-vine`.

End-user Tools

In support of the phylogeny standard and for our own research purposes, we have developed data-processing scripts and visualizations. These tools require users to provide standard-compliant input files (via a command line or graphical interface), processing the input as part of a data processing pipeline or producing a visualization of the given data.

Phylogeny Web Visualization Visualization is a critical part of data analysis, helping us build intuitions and communicate our results. We are actively developing a web-based phylogeny visualization tool that takes standard-compliant asexual phylogeny data as input and generates a phylogenetic tree, color-coded based on a user-specified numeric property (see Figure 2 for example output). Refer to our GitHub repository for more detailed usage information for this tool (`https://github.com/emilydolson/lineage_viz_tool`).

Time to Coalescence Command Line Tool The time to coalescence command line tool takes a standard phylogeny as input calculates how far back in time we need to look to find the most recent common ancestor of all extant taxa. The tool searches the given standard phylogeny file for the organisms with the greatest origin_time. It then traces the ancestors recursively until it finds the most recent common ancestor (MRCA), at which point the tool returns the time to coalescence and the id value of the MRCA. See our GitHub repository for more detailed usage information for this tool (`https://github.com/alife-data-standards/tools-pack-phylogeny`).

Conclusions and Future Directions

In this work, we discussed the motivation and vision for a set of ALife data standards. Additionally, we proposed a standard for describing and storing phylogeny data and presented several software tools that have already been devel-

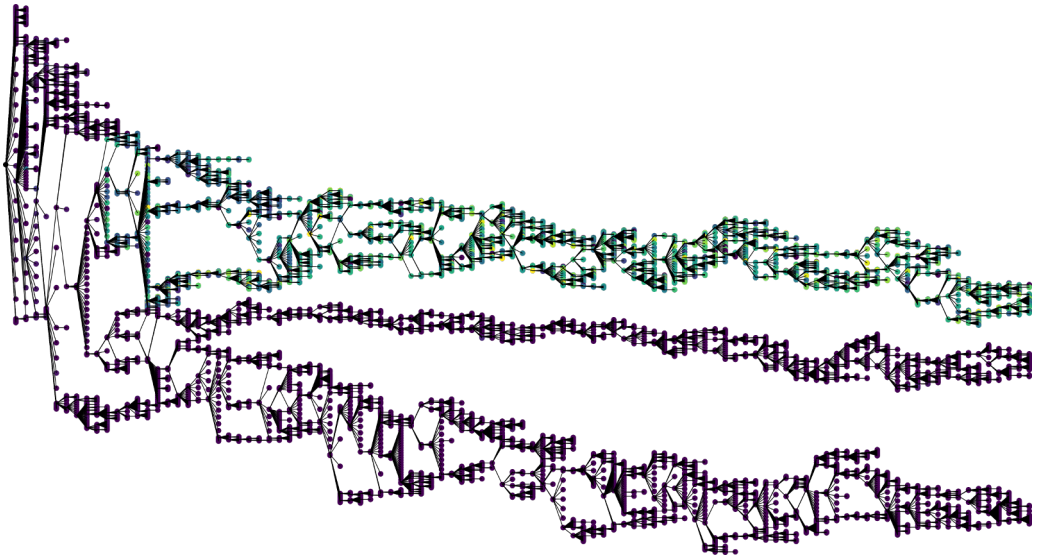


Figure 2: Example output generated using the phylogeny web visualizer. Generations proceed from left to right with the extant taxa shown on the far right. Nodes can be color-coded; here green and yellow indicate relatively high numeric values for the specified property.

oped to support the standard. We see this work as a continuation of the conversation started at the discussion-driven workshops at the 2018 Conference for Artificial Life and the 2018 Congress for the BEACON Center for the Study of Evolution in Action. By reaching out to the wider ALife community, we hope to broaden the scope of our standards, continue developing software tools that work with standardized data, and build community support for adopting and improving data standards.

Ultimately, the success our proposed data standards will depend on the level of community buy-in and adoption. The utility of these standards will grow as more of the community adopts and contributes to our ecosystem of data standards. To ensure an inclusive environment for standards development and discussion, we have adopted a Contributor Covenant code of conduct (Covenant, 2014).

As we only represent a subset of the ALife community, we do not know the full set of data standards that would be valuable to the community; for this, we turn outward: what types of data should we develop standards for? In workshop discussions, we identified the following targets for future data standards: genomes, interaction networks, fitness landscapes, and meta-data. While genomes can be broadly defined as heritable and mutable material, developing a genome standard has proven elusive because of the enormous variety of genetic representations used across different systems. Any adopted genome standard should be flexible enough to support the varied types of both artificial and natural genomes; this would allow us to make direct comparisons between digital and biological systems and make the tools we develop useful to biologists. Interaction networks describe the relationships between interacting entities (*e.g.*, objects, individual organisms, chemicals, *etc*.); for example, a food web is a type of interaction network, describing the predator-prey relationships among species represented in the network. A fitness landscape characterizes the mapping between the space of possible genotypes (or a set of phenotypic traits) and fitness for a given environment. For example, given a genome and an environment, the fitnesses of all possible one-step mutants describes the local fitness landscape adjacent to the given genome. A fitness landscape standard would allow researchers to more effectively compare fitness landscapes across multiple environments and better study how populations move through a fitness landscape over the course of evolution. Meta-data provide context for other data; for example, meta-data might identify the system or the parameters used to generate a data set (*e.g.*, a phylogeny). A standard for meta-data would facilitate improved data annotation and documentation, allowing researchers to more easily replicate experiments from other research groups.

Acknowledgements

We thank workshop participants at ALife 2018 and the 2018 Congress for the BEACON Center for the Study of Evolution in Action for thoughtful discussion and feedback on developing software standards for the Artificial Life community. This work was supported by the National Science Foundation (NSF) through the BEACON Center (Cooperative Agreement DBI-0939454), Graduate Research Fellowships to ED and AL (Grant No. DGE-1424871), and NSF Grant No. DEB-1655715 to CO.

References

Allaire, J., Ushey, K., and Tang, Y. (2018). *reticulate: Interface to 'Python'*. R package version 1.7.

Board, M. I. (1999). Mars climate orbiter mishap investigation board phase i report november 10, 1999.

Bohm, C. and Hintze, A. (2017). Mabe (modular agent based evolver): A framework for digital evolution research. In *ALife Conference Proceedings 14*, pages 76–83. MIT Press.

Bohm, C., Lalejini, A., Dolson, E., and Ferguson, A. (2019). Software Resources for the Artificial Life Data Standards. DOI: 10.5281/zenodo.2595536 URL: https://github.com/alife-data-standards/alife-data-tools.

Cardona, G., Rosselló, F., and Valiente, G. (2008). Extended Newick: it is time for a standard representation of phylogenetic networks. *BMC Bioinformatics*, 9(1):532.

Covenant, C. (2014). A code of conduct for open source projects. *Coraline Ada Ehmke*. Available at: https://www.contributor-covenant.org/.

Cranston, K., Harmon, L. J., O'Leary, M. A., and Lisle, C. (2014). Best Practices for Data Sharing in Phylogenetic Research. *PLOS Currents Tree of Life*.

Darwin Core task group, B. I. S. T. (2014). Darwin Core: 2014-11-08. DOI: 10.5281/zenodo.12694.

Dolson, E., Wiser, M. J., and Ofria, C. A. (2016). The Effects of Evolution and Spatial Structure on Diversity in Biological Reserves. In *Artificial Life XV: Proceedings of the Fifteenth International Conference on Artificial Life*, pages 434–440, Cancun, Mexico. MIT Press.

Duchemin, W., Gence, G., Chifolleau, A.-M. A., Arvestad, L., Bansal, M. S., Berry, V., Boussau, B., Chevenet, F., Comte, N., Davín, A. A., Dessimoz, C., Dylus, D., Hasic, D., Mallo, D., Planel, R., Posada, D., Scornavacca, C., Szöllősi, G., Zhang, L., Tannier, É., and Daubin, V. (2018). RecPhyloXML: a format for reconciled gene trees. *Bioinformatics*.

Gleeson, P., Crook, S., Cannon, R. C., Hines, M. L., Billings, G. O., Farinella, M., Morse, T. M., Davison, A. P., Ray, S., Bhalla, U. S., et al. (2010). Neuroml: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS computational biology*, 6(6):e1000815.

Goldsby, H. J., Dornhaus, A., Kerr, B., and Ofria, C. (2012). Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proceedings of the National Academy of Sciences*, 109(34):13686–13691.

Guelton, S., Brunet, P., Amini, M., Merlini, A., Corbillon, X., and Raynaud, A. (2015). Pythran: Enabling static optimization of scientific python programs. *Computational Science & Discovery*, 8(1):014001.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Han, M. V. and Zmasek, C. M. (2009). phyloXML: XML for evolutionary biology and comparative genomics. *BMC Bioinformatics*, 10(1):356.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., et al. (2003). The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.

Lalejini, A. and Dolson, E. (2019). 2018 standards workshop description and notes. DOI: 10.5281/zenodo.2592691 URL: https://github.com/alife-data-standards/ALIFE2018-Standards-Workshop.

Lalejini, A., Dolson, E., Ferguson, A., Bohm, C., and Rainford, P. F. (2019). Alife data standards. Version 1.0-alpha. DOI: 10.5281/zenodo.2577410. URL: https://github.com/alife-data-standards/alife-data-standards.

Maddison, D. R., Swofford, D. L., and Maddison, W. P. (1997). Nexus: An Extensible File Format for Systematic Information. *Systematic Biology*, 46(4):590–621.

Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229.

Parr, C. S., Guralnick, R., Cellinese, N., and Page, R. D. (2012). Evolutionary informatics: unifying knowledge about the diversity of life. *Trends in Ecology & Evolution*, 27(2):94–103.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Richmond, P., Cope, A., Gurney, K., and Allerton, D. J. (2014). From model specification to simulation of biologically constrained networks of spiking neurons. *Neuroinformatics*, 12(2):307–323.

Smith, B. and Welty, C. (2001). FOIS introduction. In *Proceedings of the international conference on Formal Ontology in Information Systems - FOIS '01*, volume 2001, pages .3–.9, New York, New York, USA. ACM Press.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv:1712.06567 [cs]*. arXiv: 1712.06567.

Taylor, T., Auerbach, J. E., Bongard, J., Clune, J., Hickinbotham, S., Ofria, C., Oka, M., Risi, S., Stanley, K. O., and Yosinski, J. (2016). WebAL Comes of Age: A Review of the First 21 Years of Artificial Life on the Web. *Artificial Life*, 22(3):364–407.

Wang, R., Clune, J., and Stanley, K. O. (2018). Vine: an open source interactive data visualization tool for neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1562–1564. ACM.

Wieczorek, J., Bloom, D., Guralnick, R., Blum, S., Dring, M., Giovanni, R., Robertson, T., and Vieglais, D. (2012). Darwin Core: An Evolving Community-Developed Biodiversity Data Standard. *PLoS ONE*, 7(1):e29715.

Wren, J. D. (2016). Bioinformatics programs are 31-fold over-represented among the highest impact scientific papers of the past two decades. *Bioinformatics*, 32(17):2686–2691.

Wu, Y., Wang, N., Kropczynski, J., and Carroll, J. M. (2017). The appropriation of GitHub for curation. *PeerJ Computer Science*, 3:e134.

Zaman, L., Meyer, J. R., Devangam, S., Bryson, D. M., Lenski, R. E., and Ofria, C. (2014). Coevolution drives the emergence of complex traits and promotes evolvability. *PLoS Biol*, 12(12):e1002023.

Zhang, Z., Bajic, V. B., Yu, J., Cheung, K.-H., and Townsend, J. P. (2011). Data integration in bioinformatics: Current efforts and challenges. In Mahdavi, M. A., editor, *Bioinformatics*, chapter 2. IntechOpen, Rijeka.

Zmasek, C. M. and Eddy, S. R. (2001). ATV: display and manipulation of annotated phylogenetic trees . *Bioinformatics*, 17(4):383–384.