Discovering a Domain Alphabet

Michael D. Schmidt
Computational Synthesis Laboratory
Cornell University
Ithaca, NY 14853, USA
mds47@cornell.edu

Hod Lipson
Computational Synthesis Laboratory
Cornell University
Ithaca, NY 14853, USA
hod.lipson@cornell.edu

ABSTRACT
A key to the success of any genetic programming process is the use of a good alphabet of atomic building blocks from which solutions can be evolved efficiently. An alphabet that is too granular may generate an unnecessarily large search space; an inappropriately coarse grained alphabet may bias or prevent finding optimal solutions. Here we introduce a method that automatically identifies a small alphabet for a problem domain. We process solutions on the complexity-optimality Pareto front of a number of sample systems and identify terms that appear significantly more frequently than merited by their size. These terms are then used as basic building blocks to solve new problems in the same problem domain. We demonstrate this process on symbolic regression for a variety of physics problems. The method discovers key terms relating to concepts such as energy and momentum. A significant performance enhancement is demonstrated when these terms are then used as basic building blocks on new physics problems. We suggest that identifying a problem-specific alphabet is key to scaling evolutionary methods to higher complexity systems.

Categories and Subject Descriptors
I.2.6 [**Symbolic and Algebraic Manipulation**]: Learning – *Knowledge acquisition*

General Terms
Algorithms

Keywords
Building Blocks, Domain Alphabet, Symbolic Regression

1. INTRODUCTION
Critical to the success of any genetic programming system is the use of a good alphabet of atomic building blocks from which solutions can be evolved efficiently. Typically, GP practitioners will choose generic building blocks based on prior domain knowledge, but this choice may have profound performance implications. An alphabet that is too granular may generate an unnecessarily large search space, while an inappropriately coarse grained alphabet may bias or even prevent finding optimal solutions. Here we investigate a method that identifies an alphabet appropriate for a specific problem domain automatically.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 1-58113-000-0/00/0004…$5.00.

As an example, consider the problem of evolving mathematical expressions that model data collected from an experimental system. If the system is mechanical, its expressions are likely to contain various combinations of trigonometric terms or kinetic energy terms. If the system is biological, then trigonometric terms are unlikely to appear at all; instead, reaction rates and chemical gradient terms such as Hill functions are likely to appear. The availability of appropriate building blocks greatly simplifies both the search space for mathematical models of more complex systems, as well as our conceptual understanding of the results [1]. A large portion of scientific inquiry has been devoted to unraveling these building blocks by hand. Here, we propose a computational method to explore and learn the language and rules of a problem domain automatically.

Any mathematical equation, or mathematical model, can be decomposed into various combinations of simpler building blocks, such as monomials or trigonometric terms. All of these building blocks are candidates for a common mathematical alphabet of other related systems. Therefore, to build a domain alphabet automatically, we must be able to both generate physically meaningful mathematical models, and be able to identify the nontrivial building blocks from these models.

We use symbolic regression and Pareto analysis to find physically meaningful mathematical models from experimental data. We are interested in finding the most accurate equation at different equation complexities; for example, finding the most accurate model that uses no more than six mathematical operations. These equations are special in the sense that they are both accurate and parsimonious [2] – often consisting of different approximations or elaborations of the physical description of the system.

We break down the models found on the symbolic regression Pareto front into individual terms and building blocks to form a list of candidates for a domain alphabet. The building blocks are extracted by iterating through all sub-trees (sub-expressions) of the equations. Many of these building blocks may not be useful for other systems, such as terms that are overfit to the data or numerical coincidences. Therefore, we need a way to discriminate among the various building blocks.

In order to determine which building blocks generalize to other systems in a scientific domain, we need to compare models in two or more systems (Fig. 1a). We repeat the automated modeling and Pareto analysis to generate candidate building block lists for multiple systems (Fig 1b). Finally, we calculate the frequencies that each building block is used in a different system. By considering the frequency and the complexity of a building block itself, we distill the nontrivial building blocks that are the most ubiquitous to return the alphabet of the domain (Fig. 1c).

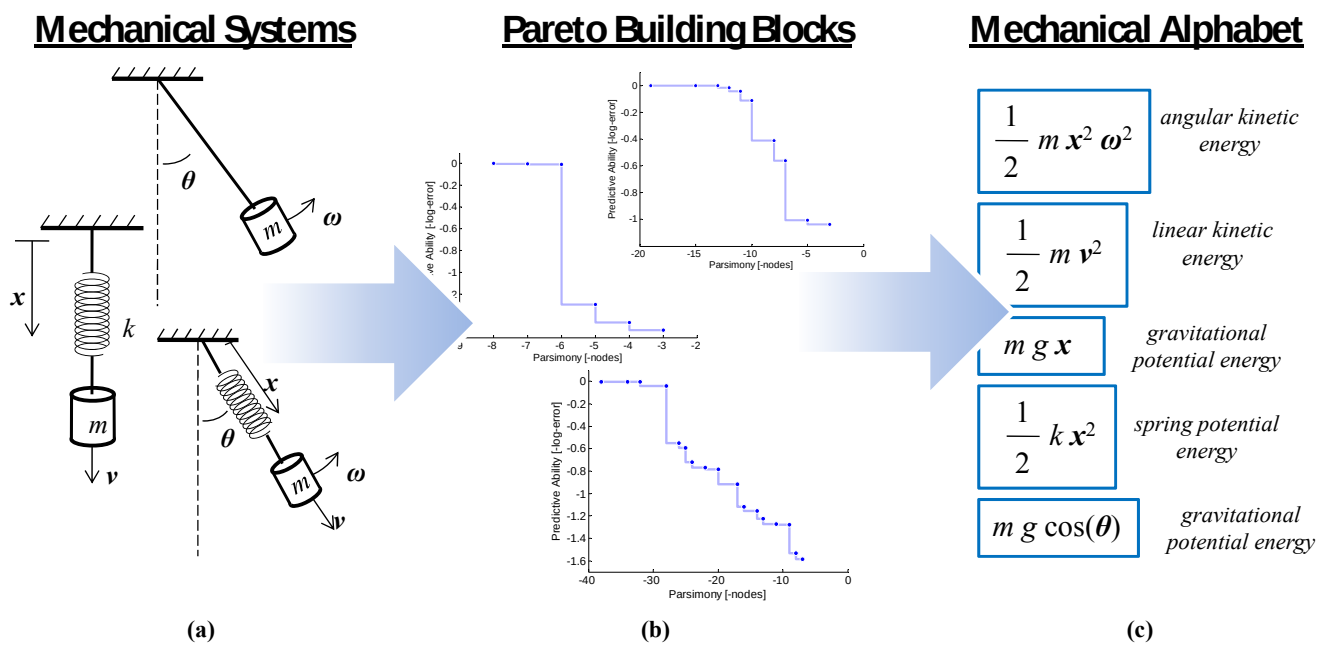


Figure 1. We distill the common mathematical language needed to describe a group of systems using symbolic regression and analysis of their model accuracy/complexity Pareto fronts. We generate experimental data from several related systems such as spring and mass mechanical systems (left). We then use symbolic regression to find several accurate models at varying complexity of equations (middle). Finally, we discompose models on these fronts to individual terms and building blocks. The most frequenly used terms and building blocks form an emergent alphabet for describing models of this group of systems (right).

In the remaining sections, we describe the background information in building blocks, accuracy/complexity Pareto analysis, and symbolic regression, followed by a detailed description of our method, experiments, and results.

2. BACKGROUND

2.1 Genetic Building Blocks

Building blocks [3, 4] are simple expressions which comprise a more complicated solution. While building blocks are most commonly associated with genetic algorithms, they can also refer to sub-trees in genetic programs [5, 6]. For example in symbolic regression (see Section 2.4), the lowest level building blocks are typically algebraic operations such as add, subtract, multiply, and divide. However, we can also define higher order building blocks such as squaring and multiplying with a constant.

We think of a solution, or equation, as being composed of various types of building blocks [7]. For example, if we think of an equation as a binary parse tree of mathematical operations (Fig. 2), the set of building blocks for that particular equation contains all sub-trees (sub-expressions) of the tree [5].

Knowing the building blocks for a particular problem simplifies human conceptual understanding of the problem (and related problems) by giving higher order meanings and interpretations of the system's mechanics, morphology, or physics. For example, rather than working with cosine operations and a set of variables, a cosine of an angle building block could allow us to work instead with a more meaningful concept, such as the vertical position of a swinging pendulum.

Knowing the basic building blocks of a system ahead of time also greatly simplifies searching for or building a mathematical model to explain its behavior and experimental data – such as done in symbolic regression. Rather than having to re-derive common terms from scratch, over and over again for each model, the algorithm could benefit from the coarser search of assembling higher order building blocks. There are an infinite number of potential building blocks however.

2.2 Domain Alphabet

While there are an infinite number of possible building blocks for any system, we define a *domain alphabet* as the set of building blocks specific to a particular problem, domain, or class of systems that generalize to many similar systems. Domain alphabet building blocks are typically physically meaningful, and are useful for building new models.

Determining the most useful building blocks can be considered to fall under the "credit assignment" problem in machine learning. The credit assignment problem is the task of deciding how to score or weight the importance of individual components of a model when only given entire systems [8]. For example, what is the importance of the gears of a bicycle for riding quickly? This clearly depends on the other components of bicycle, such as the gear ratio, the wheels, etc.

The major challenge to detecting a meaningful or useful building block is that the building block may arise by chance due to overfitting the data, or the building block may simply be a numerical coincidence. For example, consider the following equations for two different systems:

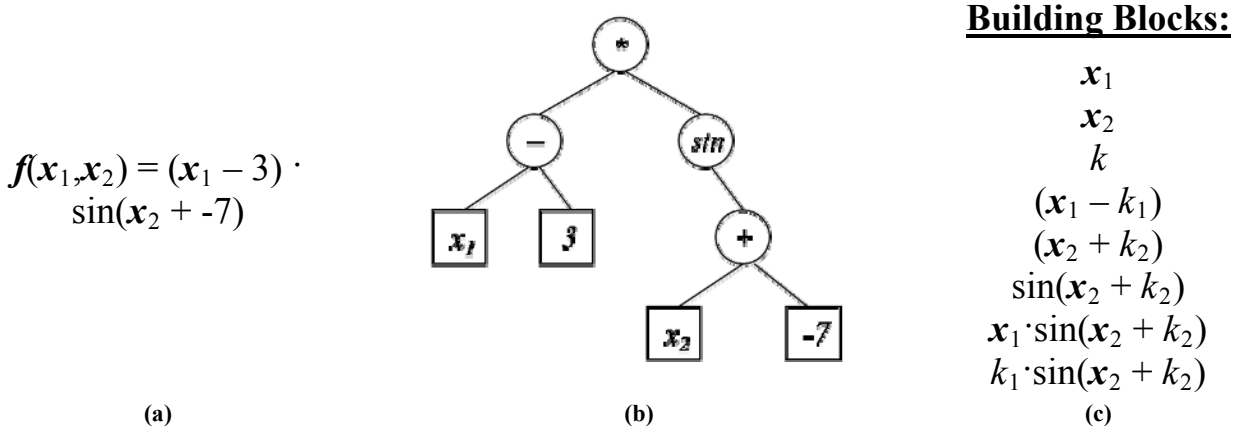


Figure 2. Example equation (a), its binary parse tree (b), and all possible building blocks of the equation (c). Building blocks are small models of internal components of a system that simplify building a full mathematical model.

$$f = x^2 \cos(x - 1.01) + 2x^3$$

$$g = x^2 \cos(x - 1.02) - \sin(x) + x$$

We would like to be able to identify a term such as $x^2 \cos(x - 1)$ as a building block given only f, g, and x values over time (we don't know the equations in advance) – while rejecting others that are less commonly generated during modeling. The more systems we look at, the less and less likely such a complex building block we be rediscovered repeatedly by chance during evolution. Therefore, finding large repeated building blocks is a strong indication the building block is a nontrival building block useful throughout the problem domain.

With such information on useful physical terms, the algorithm could reuse them for analyzing future systems, bootstrapping its knowledge into higher complexity systems. Rather than needing to rediscover common features repeatedly, the algorithm can simplify the problem to the assembly of solutions within the domain alphabet.

2.3 Pareto Front

When generating potential building blocks, we consider the *Pareto front* [9, 10] produced by symbolic regression which represents the tradeoff between a model's complexity and its maximum predictive ability for the experimental data. Parsimony is measured as the inverse of number of terms in the expression, while the predictive accuracy is measured as error on unseen data.

For any finite set of experimental data, there is potentially an infinite set of equations that maximize any type of error metric. For example, a high order polynomial equation can perfectly fit any dataset with fewer unique data points than the order of the polynomial. Therefore, it is important to have some qualitative understanding of what the domain of equations looks like.

If we consider the relationship between equation complexity and accuracy of fitting the experimental data, there are there two qualitative extremes: extremely complex equations (eg. Taylor series, neural networks, and Fourier series) with near perfect accuracy and simple, single-parameter models with baseline accuracy. The equations in-between these two extremes are the most challenging to find and identify, and their behavior is more interesting [2].

At certain minimum complexities, the predictive ability tends to jump dramatically and then plateau. In other words, there is often a relatively simple model or equation that captures some intrinsic relationships of the system (but perhaps not perfectly). By parsimony arguments, we can reason simpler equations to likely be approximations and more complex equations to be more precise refinements and elaborations of the exact model or overfit solutions to the data.

Though we can't know with certainty what the exact physical model is, it is likely to exist at least partially on this Pareto front. Therefore, when detecting what building blocks may form a general physical alphabet, we consider all building blocks on the Pareto front as candidates for inclusion in the alphabet.

2.4 Symbolic Regression

Symbolic regression [11] is a type of genetic program for searching the space of expressions computationally by minimizing various error metrics. Both the *parameters* and the *form* of the equation are subject to search. In symbolic regression, many initially random symbolic equations compete to model experimental data in the most parsimonious way. It forms new equations by recombining previous equations and probabilistically varying their sub-expressions. The algorithm retains equations that model the experimental data well while abandoning unpromising solutions. After an equation reaches a desired level of accuracy, the algorithm terminates, returning its most parsimonious equation that is most likely to correspond to the intrinsic mechanisms of the observed system.

In symbolic regression, the *genotype* or *encoding* represents symbolic expressions in computer memory. Often, the genotype is a binary tree of algebraic operations with numerical constants and symbolic variables at its leaves [12, 13]. Other encodings include acyclic graphs [14] and tree-adjunct grammars [15]. The *fitness* of a particular genotype (a candidate equation) is a numerical

measure of how well it fits the data, such as the equation's correlation or squared-error with respect to the experimental data.

A point mutation can randomly change the type of the floating-point operation (for example, flipping an add operation to a multiply or an add to an system variable), or randomly change the parameter constant associated with that operation (if it is used). The crossover operation recombines two existing equations to form a new equation. To perform crossover, we select a random location in the genotype, and copy all operation and parameter values to the left of this point from the first parent and remaining operations and parameters to the right from the second parent.

3. OUR APPROACH

Our goal is to identify the primary mathematical building blocks of a particular problem or domain of systems, thereby building a domain alphabet automatically from experimentally collected data. Our primary challenge is distilling the nontrivial building blocks that generalize to other physical systems for inclusion in the domain alphabet.

Our method has three main steps: (1) finding several mathematical models for two or more related systems, (2) decomposing these models into their constituent building blocks, and (3) identifying the most useful and meaningful building blocks for inclusion in the domain alphabet.

3.1 Modeling Groups of Systems

Our first task is to find several system models that define many candidate building blocks. We collect data from several related physical systems (Fig. 1a) by observing their behavior and dynamics over time. The group of systems should represent qualitatively different dynamics within the same problem domain.

Next, we employ a symbolic regression algorithm [16] to generate several hypothesized mathematical models of each system for varying model complexities.

The output of our symbolic regression algorithm is a small set of equations that lie on the equation accuracy and equation complexity Pareto front for each particular system (Fig. 1b). The equations on this front are nontrivial in the sense that they represent the maximum accuracy a model of a given size or complexity can achieve to explain the system's data. The equations on the Pareto front are often different levels of approximation or elaborations of the exact physical system

3.2 Extracting Building Blocks

Now that we have several equations modeling each system, we decompose them into building blocks. For each equation found on each system's Pareto front, we iterate through every sub-tree (or sub-expression) of the equation, adding the sub-tree to our list of potential building blocks (Fig 2).

During this process, we abstract away the bulk constants found in each equation and sub-expression to symbolic parameters. For example, we would convert a sub-expression such as $x + 1.427 \cdot \cos(\theta)$ to $k_1 \cdot x + k_2 \cdot \cos(\theta)$. This allows us to later match building blocks between different systems that may only vary by their embedded coefficients.

Additionally, we abstract away variable types based on their units. For example, we consider variables of angles to be equivalent to variables of lengths, but not equivalent to velocities. This allows us later to match building blocks between systems with differences in variable names.

3.3 Distilling the Domain Alphabet

We now have a long list of all building blocks found for each system and must distill this list down to a domain alphabet. We need to identify which are the nontrivial and meaningful building blocks within this list.

If a particular building block exists repeatedly on the Pareto fronts of other systems, it is a strong indication that it is a meaningful building block for the domain alphabet. At the very least, the building block is certainly useful for forming a parsimonious model in more than one system. This observation forms the basis for identifying the domain alphabet.

If a building block was simply a result of overfitting to the data, it is unlikely to be repeated on the Pareto front of other systems or different datasets because overfit solutions are very sensitive to perturbations and noise in the data. Similarly, if the building block is the result of a numerical coincidence for modeling a particular dataset, it is unlikely that the same coincidence exists in other systems and in their datasets.

Therefore, we can use the frequency that a building block is used on the Pareto fronts as a principle for its generality and importance for a domain of systems. To do this we iterate through all building blocks and count their total occurrences on the Pareto fronts of every other system, and number of times each building block was matched by another.

We form the initial alphabet by rejecting all building blocks that have zero frequency on the Pareto fronts of the other systems.

The second criterion we can use to gauge the importance of a candidate building block is its complexity. Very complex building blocks are much less likely to reoccur by chance or numerical coincidence than simple building blocks. Therefore, we also consider the complexity of the building block when adding it to the domain alphabet.

After calculating the frequencies and complexities of all potential building blocks we examine them graphically to verify the results. We plot each building block as a point on a second type of Pareto space: the building block frequency versus the building block complexity. As we discovery later, the building blocks on this chart that are both complex and frequently used comprise the physically meaningful domain alphabet.

4. EXPERIMENTS

4.1 The Mechanical Systems

We explore the alphabet building approach using a few simple mechanical systems: a harmonic oscillator, a simple pendulum, and a 2D spring pendulum. These systems are known to have well-defined mathematical models, allowing us to generate data and verify our results. Schematic diagrams of these systems are shown in Fig. 3

The harmonic oscillator (Fig. 3) is a simple conservative system with one degree of freedom. The variables are the mass's vertical position over time and vertical velocity over time. The symbolic regression algorithm identifies several equations modeling the system's kinetic and potential energy over time on the accuracy/complexity Pareto front, including the system's exact Hamiltonian equation.

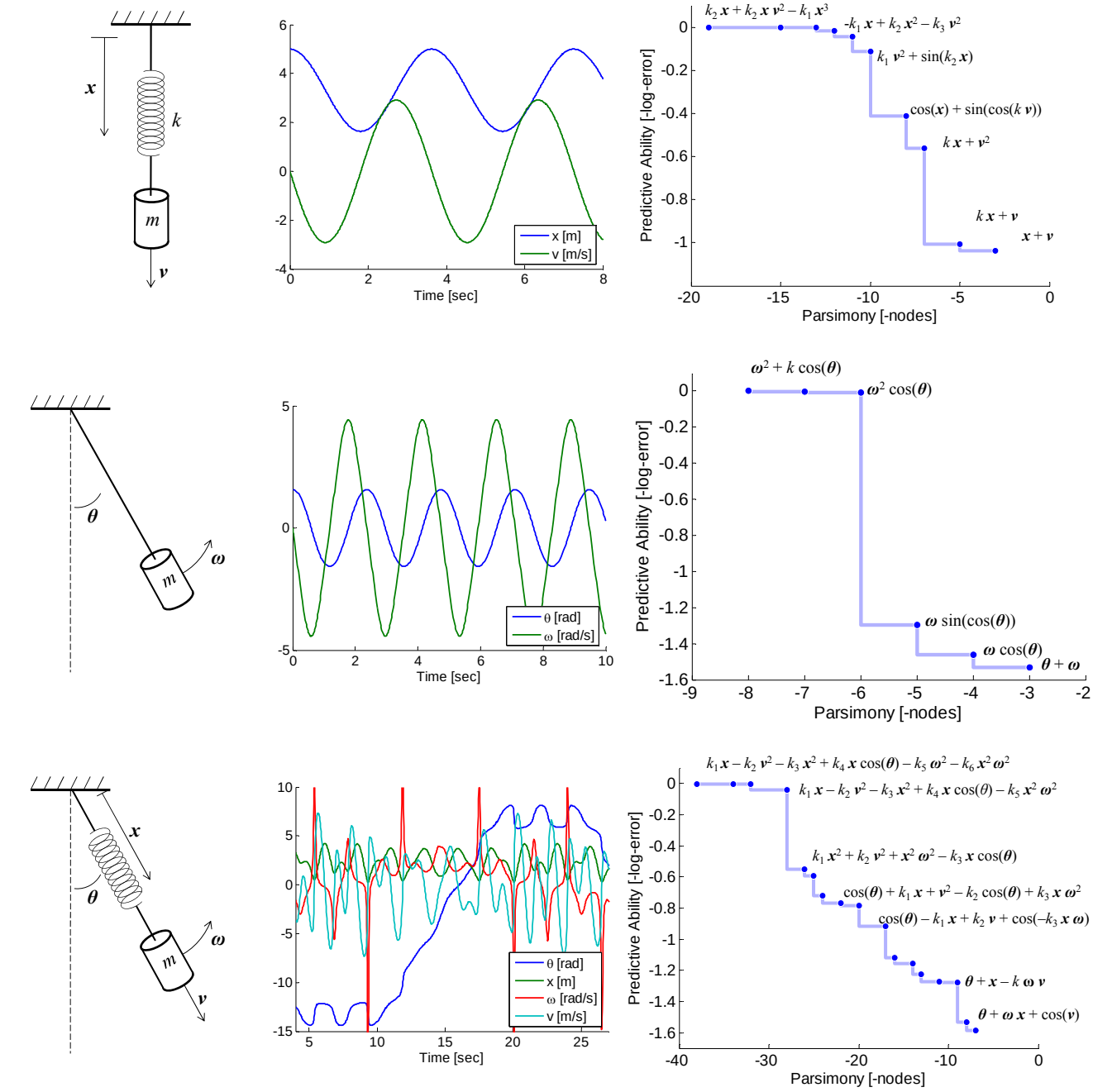


Figure 3. Summary of the mechanical systems, the collected data of their dynamics, and the resulting models found using symbolic regression on the equation accuracy and complexity Pareto front. Each system was simulated numerically. The symbolic regression algorithm generates a small set of equations for each system. This set is a Pareto front, showing the most accurate equation found for different sizes (complexities) of equations. These equations are used to distill a common mathematical alphabet of building blocks for modeling mass, spring, and pendulum mechanical devices.

The simple pendulum (Fig. 3) is a similar system, but with nonlinear trigonometric terms. The mass's position is measured by the pendulum's angle, and the velocity is the pendulum's angular velocity. Symbolic regression identifies several equations modeling the angular energies over time.

The third system is the more complex 2D spring pendulum (Fig. 3). Here, the system has two degrees of freedom, two positions, and two velocities measured over time. The dynamics of this systems are more complex, but still tractable with the symbolic regression algorithm.

The Pareto front of these systems (shown in Fig. 3) summarizes the equations that maximize parsimony and accuracy for modeling the experimental data. The terms in these equations are in this sense useful, and may comprise a common physical language.

We simulated these systems numerically by integrating their differential equations. We save the position coordinates and the velocities of each component of the system as the experimental data for use in the symbolic regression algorithm (in Fig. 3).

4.2 Experimental Setup

We used the fitness prediction symbolic regression algorithm described in [16-18] to search the space of symbolic equations. We use the deterministic crowding selection method [19], with 1% mutation probability and 75% crossover probability. The encoding is an operation list acyclic graph with a maximum of 64 operations/nodes [14]. Single-point crossover exchanges operations in the operation list at a random split. The operation set contains addition, subtraction, multiply, sine, and cosine operations.

We limit the size of the equation to narrow our search to human-interpretable equations (equations we could fit on a piece of paper). We allowed a maximum of 64 nodes, each possibly representing five types of mathematical operations, two to four variables, or a parameter constant. Ignoring the infinite parameter space, we are effectively searching a space of roughly 10^{54} parameterized genotypes.

We distributed the symbolic regression evolution over 20 quad core computers (80 total cores) using the island distributed computation method [20, 21]. The island model partitions the population of solutions into separated smaller populations residing on each computer (or core). We migrated solutions between populations at regular intervals. Every 1,000 generations (averaged over all populations), we randomly shuffle all equations among random pairs of populations.

The fitness predictor population contains 1280 predictors, distributed over 80 cores. The fitness predictor subset size is 128 indices to the full training data set. Predictors were evolved using deterministic crowding, with 10% mutation and 50% crossover.

Our fitness calculation rewards equations for modeling the systems kinetic and potential energies as described in [22] as measured over the dataset. The predicted fitness values only calculate over the smaller subset of a fitness predictor rather than the entire data set.

5. RESULTS

5.1 A Mechanical Alphabet

After building the equation accuracy/complexity Pareto fronts for each system using symbolic regression and decomposing the building blocks for each equation in each system, we plot the frequency of each building block versus its complexity (Fig. 4).

We can see that single variable terms are the most common building blocks, as well as being the simplest possible building blocks. Not shown in Fig. 4 are the numerous build blocks that were only found within a single system having zero frequency.

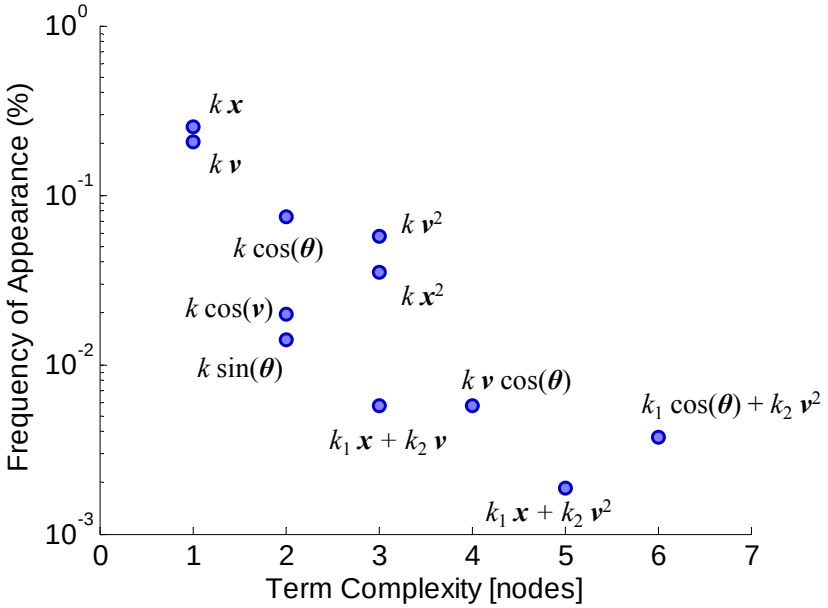


Figure 4. The building blocks found for the domain alphabet based on the harmonic oscillator, simple pendulum, and 2D spring pendulum Pareto front models. The most frequent and complex building blocks correspond to the kinetic energy terms for moving masses and potential energy terms for springs and pendula. Building blocks with zero frequency on the Pareto fronts of other systems are omitting and not included in the alphabet.

Moving to the next most frequent building blocks, we find $\cos(\theta)$ and $k\ x_2$. These are pendulum and spring potential energies respectively.

Interestingly, the higher complexity building blocks in Fig. 4 are the result of matches between inexact equations between the different systems. For example, $k_1\cos(\theta) + k_2 v^2$ is an exact building block for the simple pendulum system, but also an approximate solution to the harmonic oscillator.

There are two building blocks in Fig. 4 which are not exact building blocks for any of the systems, though they are potentially useful approximate building blocks; namely, $k\cos(v)$ and $k\sin(\theta)$. The $k\cos(v)$ term approximates a kinetic energy and the $k\sin(\theta)$ term approximates a single variable term. These terms are both low complexity and low frequency however. This hints that these are approximate building blocks and we could elect to reject them after manual inspection.

This result suggests that the terms that are both frequently used and complex tend to be more physically meaningful for inclusion in the domain alphabet, such as trigonometric terms representing potential energies or squared velocities representing kinetic energies.

5.2 Utilizing the Alphabet

One application of the domain alphabet is to simplify the search for forming models of more complex systems. We demonstrate this idea by using an alphabet formed from just the harmonic oscillator and simple pendulum systems to find a model of the more complex 2D spring pendulum system.

If we repeat the Pareto building block analysis, but now only with the harmonic oscillator and simple pendulum building blocks we obtain the building blocks shown in Fig. 5a.

There are many ways we could utilize these building blocks in the symbolic regression algorithm. We could introduce them as new functions in the operator set. Alternatively, we could seed the initial population using random combinations of these building blocks.

We chose to introduce a mutation operator that could mutate a sub-expression of an evolving equation to be a random building block from Fig. 5a. The constant terms in each building block, *k's*, were set to normally distributed random constants at the mutation event. This approach allows the building blocks to be consistently introduced during evolution, but also adapted if necessary. In the case that an alphabet building block is approximate, the evolution can still benefit from using it early on, and adapt its structure later to fit the exact system model.

There are likely much better methods for utilizing the alphabet building blocks in symbolic regression as well as other types of expert knowledge. For the scope of this paper, we want to show the proof of concept using a simple modification to the program.

Fig. 5b compares the symbolic regression of the 2D spring pendulum over time with and without the building block alphabet obtained from the harmonic oscillator and simple pendulum. The fitness is shown versus the number of function evaluations, averaged over ten independent trials.

Using the alphabet substantially improves performance over time, converging sooner onto the exact 2D spring pendulum model. The time to convergence is four times faster using the alphabet building blocks.

This result shows that an alphabet obtained from two simpler systems can be used to accelerate the modeling of a more complex system using symbolic regression.

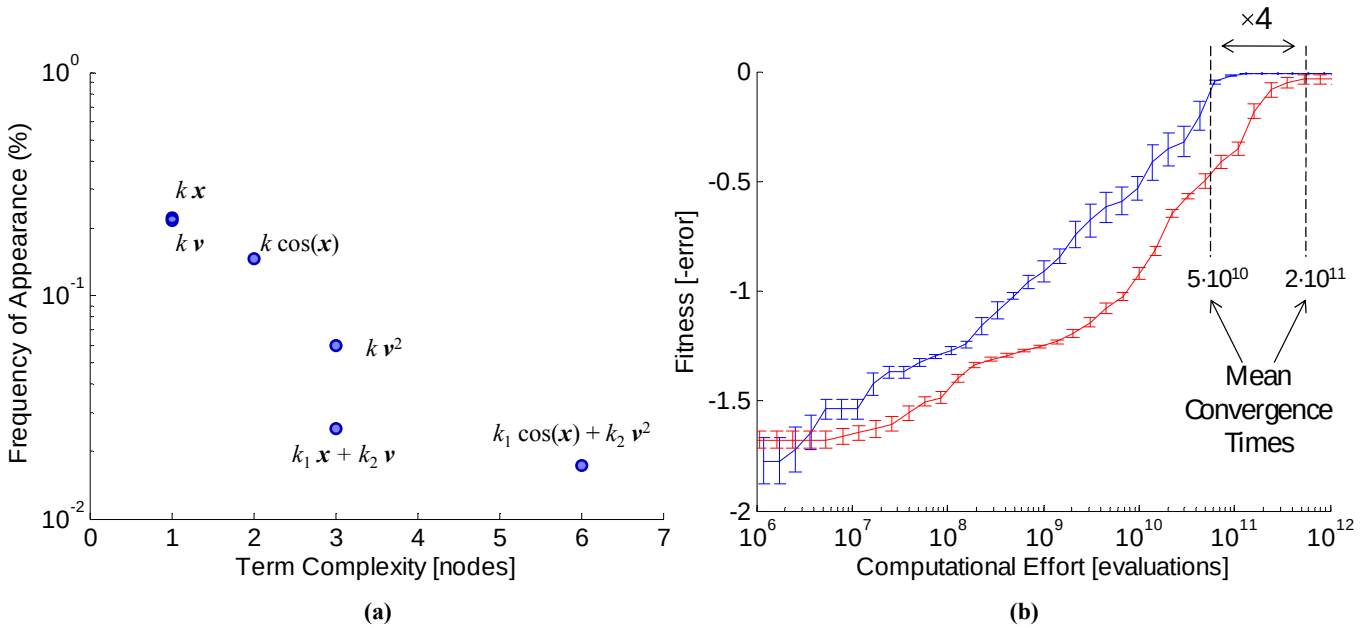


Figure 5. The impact of using a domain alphabet obtained from simple systems, the harmonic oscillator and simple pendulum, to find the model of a more complex system, the 2D spring pendulum. The alphabet in (a) shows the common building blocks found from the Pareto analysis of only the harmonic oscillator and simple pendulum systems. Allowing symbolic regression to use these terms substantially accelerates the modeling of the more complex 2D spring pendulum system (b). Error bars show the first standard error about the mean over ten independent trials.

6. CONCLUSIONS

Identifying a mathematical alphabet is a means to organize and learn the rules and language of a particular scientific field or domain. Alphabets are sets of mathematical building blocks that represent common terms and calculations that pervade different phenomena. Identifying these building blocks helps to generalize our understanding of different systems, and potentially simply the modeling of future complex systems.

We proposed an automated method to distill the mathematical alphabet directly from experimental data using symbolic regression. The method finds a set of equations for multiple related systems on the accuracy/complexity Pareto front, decomposes these equations into building blocks, and then calculates the frequencies these building blocks occur on the Pareto fronts of the other systems.

Our results suggested that building blocks that are both frequently used and complex tend to be the most physically meaningful to the class of systems; such as spring potentials and kinetic energies. Other building blocks in the resulting alphabet were potentially useful approximations common across multiple systems, such as small angle approximations, but were the least complex and least frequently used.

Finally, we used an alphabet obtained automatically from the harmonic oscillator and simple pendulum systems to accelerate the symbolic regression of the more complex 2D spring pendulum system. The regression using the alphabet found the exact model in one fourth of the computational effort compared to the regression from scratch, suggesting an automated method for scaling into higher and higher complexity systems.

7. ACKNOWLEDGMENTS

This research is supported by the U.S. National Science Foundation Graduate Research Fellowship Program, and U.S. National Science Foundation Grant CMMI 0547376.

8. REFERENCES

[1] J. H. Holland, "Building Blocks, Cohort Genetic Algorithms, and Hyperplane-Defined Functions," *Evolutionary Computation,* vol. 8, pp. 373-391, 2000.

[2] M. Kotanchek, G. Smits, and E. Vladislavleva, "Trustable symbolic regression models: using ensembles, interval arithmetic and pareto fronts to develop robust and trust-aware models," in *Genetic Programming Theory and Practice V*, 2008, pp. 201-220.

[3] J. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*: University of Michigan Press, 1975.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley Longman Publishing Co., Inc., 1989.

[5] U.-M. O'Reilly, "The Trouble Aspects of a Building Block Hypothesis for Genetic Programming," Santa Fe Institute Feb 1994.

[6] J. P. Rosca, "Towards automatic discovery of building blocks in genetic programming," in *Working Notes for the AAAI Symposium on Genetic Programming*, E. V. Siegel and J. R. Koza, Eds. MIT, Cambridge, MA, USA: AAAI 445 Burgess Drive, Menlo Park, CA 94025, USA, 1995, pp. 78-85.

[7] N. McPhee, B. Ohs, and T. Hutchison, "Semantic Building Blocks in Genetic Programming," in *Genetic Programming*, 2008, pp. 134-145.

[8] J. J. Grefenstette, "Credit assignment in rule discovery systems based on genetic algorithms," *Machine Learning,* vol. 3, pp. 225-245, 1988.

[9] C. Fonseca and P. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Genetic Algorithms: Proceedings of the Fifth International Conference*, 1993, pp. 416-423.

[10] C. M. Fonseca and P. J. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation,* vol. 3, pp. 1-16, 1995.

[11] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[12] M. Ben, J. W. Mark, and W. B. Geoffrey, "Using a Tree Structured Genetic Algorithm to Perform Symbolic Regression," in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA*. vol. 414, A. M. S. Zalzala, Ed. Sheffield, UK: IEE London, UK, 1995, pp. 487--492.

[13] D. Edwin and B. P. Jordan, "Multi-Objective Methods for Tree Size Control," in *Genetic Programming and Evolvable Machines*. vol. 4, 2003, pp. 211--233.

[14] M. Schmidt and H. Lipson, "Comparison of tree and graph encodings as function of problem complexity," in *Proceedings of the Genetic and Evolutionary Computation Conference*, London, 2007, pp. 1674--1679.

[15] X. H. Nguyen, R. I. McKay, and D. L. Essam, "Solving the Symbolic Regression Problem with Tree-Adjunct Grammar Guided Genetic Programming: The Comparative Results," in *The Australian Journal of Intelligent Information Processing Systems*. vol. 7, 2001, pp. 114--121.

[16] M. D. Schmidt and H. Lipson, "Coevolution of Fitness Predictors," *IEEE Transactions on Evolutionary Computation,* vol. 12, pp. 736-749, Dec 2008.

[17] M. D. Schmidt and H. Lipson, "Coevolution of Fitness Maximizers and Fitness Predictors" in *Proceedings of the Genetic and Evolutionary Computation Conference, Late Breaking Paper*, 2005.

[18] M. D. Schmidt and H. Lipson, "Co-evolving Fitness Predictors for Accelerating and Reducing Evaluations," in *Genetic Programming Theory and Practice IV*. vol. 5, L. R. Rick, S. Terence, and W. Bill, Eds. Ann Arbor: Springer, 2006, pp. 113-130.

[19] S. W. Mahfoud, "Niching methods for genetic algorithms," University of Illinois at Urbana-Champaign, 1995.

[20] F. Francisco, S. Giandomenico, T. Marco, and V. Leonardo, "Parallel Genetic Programming," in *Parallel Metaheuristics*, A. Enrique, Ed. Hoboken, New Jersey, USA: Wiley-Interscience, 2005, pp. 127--153.

[21] G. Christian, P. Marc, and D. Marc, "A Robust Master-Slave Distribution Architecture for Evolutionary Computations," in *Genetic and Evolutionary Computation Conference Late Breaking Papers*, R. Bart, Ed. Chicago, USA, 2003, pp. 80--87.