Artificial Intelligence for Engineering Design, Analysis and Manufacturing

cambridge.org/aie

Research Paper

Cite this article: Whiting ME, Cagan J, LeDuc P (2018). Efficient probabilistic grammar induction for design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **32**, 177–188. https://doi.org/10.1017/S0890060417000464

Received: 4 November 2016
Revised: 14 April 2017
Accepted: 28 April 2017

Key words:
Graph grammar; grammar induction; shape grammar

Author for correspondence:
Jonathan Cagan, E-mail: cagan@cmu.edu and Philip LeDuc, E-mail: prl@andrew.cmu.edu

Efficient probabilistic grammar induction for design

Mark E. Whiting, Jonathan Cagan and Philip LeDuc

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

The use of grammars in design and analysis has been set back by the lack of automated ways to induce them from arbitrarily structured datasets. Machine translation methods provide a construct for inducing grammars from coded data which have been extended to be used for design through pre-coded design data. This work introduces a four-step process for inducing grammars from un-coded structured datasets which can constitute a wide variety of data types, including many used in the design. The method includes: (1) extracting objects from the data, (2) forming structures from objects, (3) expanding structures into rules based on frequency, and (4) finding rule similarities that lead to consolidation or abstraction. To evaluate this method, grammars are induced from generated data, architectural layouts and three-dimensional design models to demonstrate that this method offers usable grammars automatically which are functionally similar to grammars produced by hand.

Introduction

Grammars are a proven tool for the representation, generation, and analysis of design (Stiny, 1980); however, difficulty in producing grammars has hindered their widespread adoption outside of academia. The key challenges of using shape grammars – *generation*, *parsing*, and *inference* (Gips, 1999) – have been partially addressed by the literature over the past 20 years in the form of grammar interpreters (Piazzalunga & Fitzhorn, 1998; McCormack & Cagan, 2002; Trescak et al., 2009), but automated grammar generation through inference programs remains largely unsolved for design grammars (Chau et al., 2004). A foremost challenge of automatically inducting grammars from a suitable dataset is that this process is of combinatorial complexity (Yue & Krishnamurti, 2013) and thus takes impractically long. Intuitively, forming a shape or graph grammar is a process of breaking a target graph down into a manageable set of morphological rules. To simply reproduce a graph is easily feasible, but to do so while maintaining subtlety of style or meaning is difficult and to do it provably optimally requires considering every plausible rule and determining which minimal set demonstrates the grammar best, which makes it computationally intensive.

Traditionally, grammar rules have been developed by hand and utilized via grammar interpreters (Gips, 1999), which offer ways to apply grammar rules in order to achieve specific design outputs. Grammar inference, or grammar induction, requires developing a plausible set of rules, which may then be applied as needed to represent candidate designs (Gips, 1999). Interpreters either automatically (Orsborn & Cagan, 2009), or through user support, execute judgments about the coherent use of rules (Trescak et al., 2012). Inference, on the other hand, requires a relatively reversed judgment to be made, the process of determining where one rule stops and the next rule starts.

Language translation is limited by many of the same challenges as developing grammars for design (Gips, 1999). However, due to the comparative linearity of language models, approaches for learning lingual grammars have existed for some time (Berwick & Pilato, 1987). The modern approach to machine translation involves extracting a grammar and lexicon for both the starting and final languages and constructing a transform from one to the other (Sánchez-Martínez & Pérez-Ortiz, 2010; DeNero & Uszkoreit, 2011). For example, when a sentence is translated from one language to another, grammatical models are used to improve translation quality significantly. Services that provide translation features use a number of other machine-learning techniques to reduce errors and improve the speed of computation for this process (Schwenk, 2012; Mikolov et al., 2013), but in essence, the practice of building a grammar and executing it is a quintessential step in effective translation. As a result, there are various mechanisms for inducting lingual grammars and ontologies that have been made efficient enough for practical use through years of refinement by industry players (Balahur & Turchi, 2014). For example, many popular translation tools use language models that are constantly updated based on popular usage to improve accuracy and efficiency around new uses of language and jargon. Additionally, advanced methods of chunking have been developed to improve the performance of such systems (Kudo & Matsumoto, 2002; Lee & Wu 2007).

Learning from these advances in machine translation and their applications to design (Talton et al., 2012), this paper introduces a means for efficiently inducing grammars for design by using a probabilistic chunking method (Lee & Wu 2007) paired with a multi-scale random walk-based graph exploration approach (Kang et al., 2012) to accommodate the complexities required in inducing design representations. The probabilistic approach does not reduce computational efficiency in the worst case, a random graph, but drastically reduces it in most cases likely to be assessed with grammars because chunk recognition can be done at a rate that is significantly faster than exhaustive searches, and the random walk allows for many discrete explorations of the graph without a time-intensive exhaustive search in most cases (Fouss et al., 2007).

To evaluate the success of the introduced method, experiments were conducted assessing four key attributes of each graph: accuracy, variability, repeatability, and conciseness. The experiments utilized several datasets, including rule-based unidimensional graphs, rule-based multi-dimensional graphs, historic buildings with well-established hand-made shape grammar representations, and a greenhouse design with component-wise specificity. Rule-based graphs were used to demonstrate basic effectiveness, while real-world examples demonstrate specific applications of the induction method as well as alignment with human-based grammar induction results.

The remainder of this paper will outline the related work in machine translation, graph isomorphism, and grammar induction that lead to these results, the specific details of the proposed grammar induction method and its evaluation across three experiments.

Related work

The difference between building grammars for translation and building grammars for design is that in general design problems do not use one-dimensional (1D) data structures, such as those of written languages, and do not have clear representations, such as letters and words in written languages. Instead, shape grammars (Stiny & Mitchell, 1978; Stiny, 1980) use rules that refer to shape elements such as points, lines and surfaces to visually represent designs, and Graph grammars (Rozenberg, 1997; Schmidt & Cagan, 1997), which refer to sets of rules that adjust a graph's ontology introducing nodes or edges. The sophistication of grammars used in design over those of language is important in several specific ways. Most directly, more complex representations introduce new ways to parse the data to be induced into a grammar, because where in text every word is immediately preceded by at a maximum one word, and tailed by at a maximum one word, in a shape design or graph context, any element or node could be connected to a large number of other elements, and this must be accommodated to build a complete grammar.

Graph similarity

Another aspect that makes building grammars for graphs challenging is that checking if two rules are the same is computationally intensive, and becomes exponentially harder as the average degree of the graph increases. In other domains, this problem is known as a subgraph isomorphism problem (McKay & Piperno, 2014), in which the task is to efficiently compare two subgraphs and determine if they are the same graph. Figure 1 shows two graphs that are identical in terms of connection but completely different in terms of the visual arrangement.

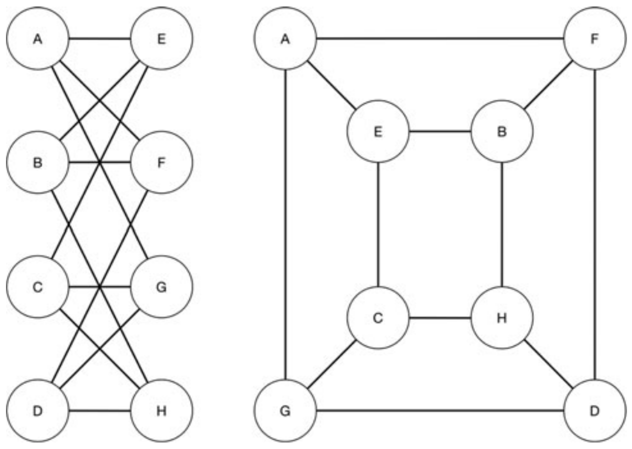


Fig. 1. Graph Isomorphism. Graphs *G* and *H* are identical sets of nodes and edges and hence isomorphic, but visually appear very different.

These graphs are isomorphic but do not appear so. Graph isomorphism is widely considered non-deterministic polynomial-time hard (NP-hard) (though recent work in this field intends to show that it may be polynomial) (Babai, 2015). The current state-of-the-art approach to computing solutions to this type of problem are of the order of $2^{O(\text{sqrt}(n \ \log(n)))}$ (Babai et al., 1983).

Humans can process higher-level similarities intuitively, so when building design grammars, they usually have the advantage of not needing to spend the time to determine if two parts of an ontology are identical using brute force (Speller et al., 2007). Computational analysis does not have this intuitive advantage, so, for example, determining if two car door moldings from opposite sides of a car are the same door or a different door requires an extensive search involving finding an orientation in which similarity and symmetry are matched and then evaluating on a per-node basis if the two parts are similar. With shape grammars, the nodes might be sets of vertexes or B-splines (Chau et al., 2004) while for graph grammar representations the nodes are some other kind of data object representation depending on the context (Rozenberg, 1997). Additionally, when considering richly connected ontologies, humans also have difficulty making intuitive sense of the similarities of graphs, as Figure 1 demonstrates.

Inducing grammars without coded data

Formulating formal grammars is a critical step in machine translation (Ding & Palmer, 2005). However, language data are full of examples of misuse, so probabilistic grammars can be used when formal representations cannot be computed. In these situations, a field of representations is used with probability weightings to generate systematic awareness of which cases are most likely to be the ones that should be formalized (Stolcke & Omohundro, 1994). Approaches like these afford inducing grammars from datasets that exhibit specific properties such as an annotated vocabulary, but cannot effectively support inducing grammars from generalized data; in other words, they do not resolve situations with no syntactical coding. As a result, they are not particularly suitable for making grammars in design more available or powerful at this time, because they would require input data with some structural coding, but may become useful as more dynamic approaches emerge. An example of this is demonstrated by Talton et al. (2012), showing that with coded design data, in which elements are functionally or stylistically categorized before evaluation, machine translation methods can be effectively used to generate grammars.

In addition to the reasons mentioned above, generating grammars automatically requires assessing many possible rules, and there are few metrics for determining the single best grammar out of a set of possible grammars (Königseder & Shea, 2015). As a result, high-level sketches for automated shape grammar induction have been proposed (Gero, 1994; Schnier & Gero, 1996; Gips 1999), and some parts of the process have been automated with statistically generated rules (Orsborn & Cagan, 2009; Talton et al., 2012). Additionally, context-sensitive graph grammar induction has been elucidated through VEGGIE (Ates & Zhang, 2007). However, these too do not sufficiently solve the issue for design grammar induction on data without semantic interpretability, which remains a major underlying challenge.

Evaluating grammar generation

In design, a grammar's effectiveness can be judged first by its ability to communicate a design's style, and then by its ability to do so in an optimal way, from a computational performance perspective. As a mechanism for conveying style, there are at least three critical previsions for effectiveness, adapted from Stiny and Mitchell (1978): (1) it should clearly convey the common elements of stylistically similar artifacts; (2) it should be complete enough to facilitate determining if an artifact is within a style or not; and (3) it should be usable to generate new artifacts that adhere to the style but were not part of the style corpus.

The *accuracy* of a grammar, defined as how well the grammar affords representing the input materials used to establish it, is intended to evaluate Stiny and Mitchell's first point above. However, quality of representation is essentially binary with non-probabilistic grammars because there is no margin of error – it is either correct or not correct – so for this work, accuracy is assumed to be a requirement of any grammar. Without this feature, the grammar is not even reliable enough to reproduce its own input and it should be rejected. Accuracy reports the percent of instances that a grammar induction system can produce its input. This is different from saying that the grammar is entirely stylistically accurate, because it can only be accurate enough to represent the data it is given, and it is important to note that accuracy is not binary if a grammar incorporates probabilistic representations, which are not focused on in this work.

The *variability* of a grammar, on the other hand, interpreting Stiny and Mitchell's third point, is defined as how well a given grammar can be used to generate new examples of the style or construction rules embedded in the grammar. Again, with non-probabilistic grammars, a grammar either offers variability, or does not, so this will be considered a necessary condition for a grammar to be accepted. Variability reports the percent of instances of a grammar induction system that can produce alternatives to their input.

Another aspect of a grammar system is the expectation that a similar grammar could be achieved from different observed artifacts. A practical example of this is if one builds a lingual grammar for English from two different instances of the New York Times Newspaper, there would be some hope that the contents of the two grammars would be similar and functionally interchangeable. We term this as the *repeatability* of a grammar, or the likelihood that grammars A and B will be identical when grammar B is learnt based on material generated from grammar A. In practice, grammar variability means that most data can be represented by many different valid grammars, so it is better to require that there is a grammar B in the set of grammars learnt from material generated by grammar A, such that grammar A and B are functionally identical. Repeatability is not exactly the same as Stiny and Mitchell's second point because for repeatability it is assumed that grammars of two different artifacts are comparable, as opposed to determining the degree to which one grammar serves two different artifacts. However, these two approaches end up being computationally similar in the worst case, because applying a grammar to an existing artifact is approximately as hard as developing a new grammar from that artifact. This work does not prove that claim, but since our definition of repeatability is stricter than Stiny and Mitchell's second point, that is the preferred metric in this work. Specifically, repeatability reports the percentage of paired rules when computing a rule set from an induced grammar output.

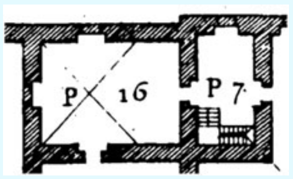
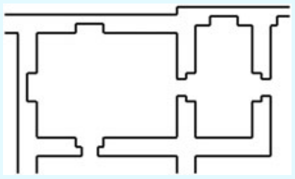
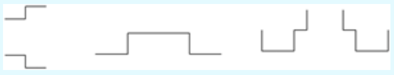
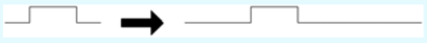
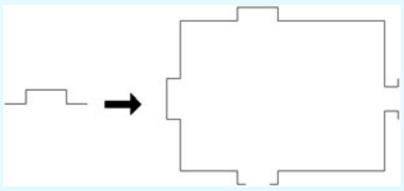
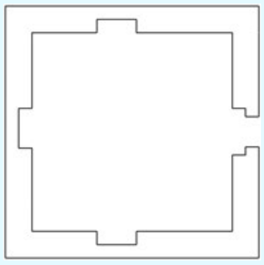
Computational complexity of grammars is a well-studied challenge (Slisenko, 1982; Yue & Krishnamurti, 2013), and determining if two different grammars can have identical output, with only a difference in complexity, is non-trivial. Prioritizing *conciseness* in generated grammars can be established by adhering to the information axiom found in Axiomatic Design (Suh, 2001); if two designs are otherwise equal, choose the simpler one. When learning grammars, and after ensuring they have accuracy, variability, and repeatability, the next priority is to establish that the selected grammar is the simplest. In practice, this is convenient for computational complexity but also because it implies that more salient information is stored per grammar rule, so arguably, it can demonstrate more nuance in alternative outputs. Conciseness in this work is reported as the percentage of rules in a grammar per unique node type in the data. Because some input data may have more types of nodes and others may have more types of rules connecting fewer nodes, it is not possible to set a specific goal for conciseness between datasets. However, it is feasible to minimize this among possible grammar solutions for a given dataset or group of similar datasets. This can be achieved by adjusting parameters such as the starting point of grammar induction or rule definition metrics, and evaluating the relative conciseness of each approach and selecting the most concise grammar representations. Additionally, across a large number of datasets, this value tends toward 50% because on average a rule represents a relationship between two node types.

Together, accuracy, variability, repeatability, and conciseness offer a multi-factor means for establishing computational tractability as well as effective communication of style. Achieving the first two is a necessary condition for a grammar to be considered effective. The latter two offer helpful insight when deciding which grammar representation best suits a given dataset. These will be used as key metrics for determining the effectiveness of grammar induction methods introduced in this paper.

Methods

The approach introduced in this paper induces arbitrary grammars from ontological data via the following steps (Table 1): (1) deconstruct the data into objects that constitute atomic units of meaning in the context of the data (e.g., words in the text, points in a drawing, people in a social network); (2) formulate structures from known relationships between objects in the original data (e.g., word pairs, lines, and friendships, respectively); (3) compute frequency-based assertions about probable interrule relationships; (4) establish new rules from probable interrule relationships that

Table 1. Grammar induction example

Data type	Text	Building plan
Raw input	…This is a sample English sentence….	
Step 1. Extracting objects	(This, is, a, sample, English, sentence, .)	
Step 2. Forming structures	(This) → (This, is) (is) → (is, a)	
Step 3. Expanding rules	(This) → (This, is, a)	
Step 4. Interrule relationships	(This) → (This, is, a, (insert object rule))	
Sample Output	This is a sample sentence.	

Demonstrating four steps and sample output based on two types of data, sample text, to show the simplest form of rule development and a building plan [a subset of the Villa Malcontenta (Rowe, 1977)], to show the application in a shape grammar context. Raw input data are segmented in step 1. Step 2 involves forming structures in the ontology. In step 3, relationships between structure sets are used to define rules. In step 4, parameterized relationships are used to form more sophisticated rules. Finally, output examples are shown that express different features from the input but also incorporate similar stylistic aspects.

were found. The resulting rule set will be concise and complete; however, it may also introduce a number of anomalous rules that can be removed by expanding the dataset and checking for coherence between elements in the ontology. Because this approach can use abstract ontological data, it can be used for both shape and graph grammars, and consequentially, the grammar representation in this work stores only ontological structure of the rules. Figure 2 shows the abstracted form of this algorithm in pseudocode, and in the following subsections these steps are discussed in detail with contextualizing examples.

Deconstructing data for grammar induction

The first step is a deconstruction of data into objects. This involves breaking a dataset down into the smallest units that are relevant to a grammar. This is done so that the dataset can be treated as a computationally addressable graph and so that patterns in that graph can be used to establish the grammar.

The process of data deconstruction is domain dependent, so approaches for deconstruction depend on the intrinsic hierarchy of the data representation, and the degree of connectedness within the data. For example, in the text, words are easily extracted producing a linear graph as is shown in step 1 in Table 1. In more complex data such as ontologies that represent buildings, data could be split into entities like a light switch or toilet, or any other element that is an isolated participant in the overall system graph. Piecewise models afford this kind of component level decomposition, so extra steps are not required for models that provide inherent hierarchy, such as the COLLADA file format (Barnes & Finch, 2008). If interpreting blueprints on the architectural feature level, points and lines can be treated as objects in this way, with positional information serving as the data structure;

```
Input: Data representing a graph of connected objects
Output: A set of rules that can be used to reconstruct the input graph
1. Create a list of objects in the graph
2. While the list is not empty:
    2.1. Remove the first object from the list and use it
    2.2. Find instances of this object in the graph
    2.3. Store collections of nodes within a distance parameter as candidate rules
    2.4. Check for collections that are structurally identical; store these as candidate inter
         rule relationships
3. Repeat 2 with the list of collections instead of objects
4. Remove duplicates
```

Fig. 2. Pseudocode of the automated grammar induction algorithm.

however, detailed positional data are not needed when treating these structures as ontologies because position information can be stored alongside the intrinsic structure of the component graph. It is important to note that decomposition in more sophisticated, integrated models, such as an A-class injection molding requires specialized methods that are not examined in this work.

Formulating initial structures

In the second step, structures are formed based on only the structure of the graph and converted into initial rule candidates. Rules take the generic form of a token on the left-hand side of the rule and a replacement token on the right-hand side of the rule, which can replace the left-hand side token in its given context. In the text, this means a particular string can be replaced by an alternate string. In graphs, rules adhere to the same formalism, identifying a subgraph that can be replaced by an alternate subgraph and rejoin with the same edges in the graph. This way of dealing with rules allows for shape or graph grammar rules to be treated relatively similarly during induction. It also allows for rules to refer to large areas or chunks, and affords meta-rules, or rules that refer to other rules as elements of the graph.

For parsing text, the words before and after each word become strong candidates for rules and they can be processed without any awareness of the part of speech or meaning of individual words. Rules can later be expanded to encompass larger groups of words or parametric patterns as they are encountered by the parser. Step 2 in Table 1 demonstrates this for the initial formulation of structures.

In complex ontologies, establishing rules is less straightforward than in text because it requires finding matching subgraphs. To do this, ideally, every traversal of the ontology would be evaluated, by enumerating possible paths through the graph, so all possible rule options are considered. This is not computationally practical, especially for heavily connected graphs, so constructing larger structures as chunks or sets of rules, by grouping or topological patterns, can be used to reduce the load (Ates & Zhang, 2007). A useful chunk is a larger region of the graph that is repeated, either identically, or thematically, in more than one place in a graph. Chunks are formed by finding matching regions in the ontology and exploring their local neighbors to find larger regions of similarity. In the example shown in step 2 of Table 1, the architectural chunks demonstrate a key characteristic of this approach, that frequently repeated elements will appear as strong chunk candidates (e.g., door and window frames), while repeated elements that have internal differences are less strong candidates (e.g., the section of wall next to one side of a window frame). Because of the recursive nature of this process, rules and chunks are ontologically similar so no cutoff is needed, and chunking can continue until no more repeated graph sections are found.

Probabilistic enhancement

The frequency of a rule being found in the dataset is interpreted as a predictor of the likelihood of its use in the final grammar, so this is the main indicator that is used to determine how to extend initial rules of pairs of objects to encompass more objects as either 1D lists or graphs. For textual data, this is a straightforward process of checking the data for repeated pairs or patterns through a direct string comparison on the rule level as is shown by step 3 in Table 1. The frequency of repetition of a pattern may be used to determine its likelihood in the graph.

For graphs, if any two rules match, the surrounding areas in the graph are evaluated and local patterns are defined. The process of frequency analysis for establishing rule likelihood is done continuously so by the time a pass of the data has been completed, both simple and complex rules (chunks) have already been established. This is necessary because the data is sampled randomly from unexplored parts of the graph, to enhance computation speed and avoid getting stuck exploring repeated localities. For well-structured, intrinsically rule-abiding datasets, it is generally the case that all rules will be found before a full traversal of the data is complete because chunks make up a majority of such data.

Rule parameterization

Interrule patterns are also established to extend the probabilistic rule set. This is functionally similar to evaluating a grammar on the rule level. The set of rules already in the grammar are treated as the input corpus for another grammar-learning process. This builds parameterized rules that can refer to a contextual variable such as a rule or token that could be used to complete some part of the graph. The new abstract rule parameterization is then added back to the initial grammar as higher level rules. Step 4 of Table 1 shows this in a text example where an object rule can be used to augment a new parameterized rule that has been developed. In practice, for text, these types of rules may represent higher-level features such as writing tropes or an author's particular writing style. In this way, this process can keep occurring providing more and more abstracted versions of rules governing a dataset, and these higher-level rules can be optionally stored separately for use in other analysis approaches, such as trying to define meta-styles in writing.

The parameterization of rich ontologies is a time-consuming process because it requires the comparison of many rules to generate higher-level rules. However, it can be executed by repeatedly running the same induction approach while new rules and their parameterizations are established (Rawson & Stahovich, 2009). This also ensures that groups of rules do not overlap significantly, which helps facilitate conciseness of the resulting grammar. This is arguably the slowest stage of the grammar induction approach but doing it after structured rules have been established reduces computation time.

Efficiency in induction

The first and second steps are fast and run in linear time with the number of rules. The third and fourth steps are non-deterministic and slower, depending on the ease of establishing equivalence between rules. In datasets where rules are unidimensional, equivalence can be done very quickly by using a hashing function on both sides of every rule. Hashing functions allow mapping large input signals, such as many lines of text, to a unique and fixed length key (Knuth, 1998). When data have more interconnection, such as in a social network or a building ontology, the hash approach only encompasses a single representation of the graph, and can lead to significant inaccuracies. The hashing approach only works on very accurate datasets and does not show any partial equivalence for noisy matches. As a result, in cases such as dealing with typos or poorly generated drawings, preprocessing of data may be required to create accurate matches to ensure objects that are semantically the same are interpreted similarly by the induction algorithm.

Instead of hashing, when inducting graphs, the isomorphism is established using the fastest current algorithm from Babai et al. (1983). Additionally, graph chunks can be hashed, affording speed increases as larger rules are constructed.

Choosing steps to improve the performance of a grammar induction process is heavily dependent on attributes of the representation and underlying structure of the data. For this reason, provisions for processing each dataset to be evaluated will be outlined specifically.

Evaluation

Evaluations were conducted on three distinct datatypes: generated patterned ontologies, a building with known grammar representations, and a structured piecewise model. Generated partnered ontologies allow analysis of the proposed methods in an abstract context and allow for ontologies exhibiting specific features to be evaluated. Many buildings have received attention from the academic community and have well-established grammatical representations which can be used for comparison. Because of the large body of existing grammatical analysis and clear rule-abiding layout, Andrea Palladio's Villa Foscari, La Malcontenta, a home near Venice, Italy built in 1550, was chosen for evaluation (Rowe, 1977; Stiny & Mitchell, 1978). The piecewise model used is a 3D model of an integrated, automated greenhouse in which each part is individually modeled and independently identified. Evaluating data of this kind allows multi-scale graph representation of the structure and facilitates highly accurate chunking. The analysis was conducted using the accuracy, variability, repeatability, and conciseness framework.

Inducing generated ontologies

Randomly generated grammar rules were used to build the artificial datasets, as opposed to purely random data, to ensure that there were sufficient patterns for the grammar induction algorithm to find.

A set of rules was constructed and then applied at random to produce datasets with specific size and connection properties. Graphs were generated with small ($n = 100$, shown in Fig. 3),

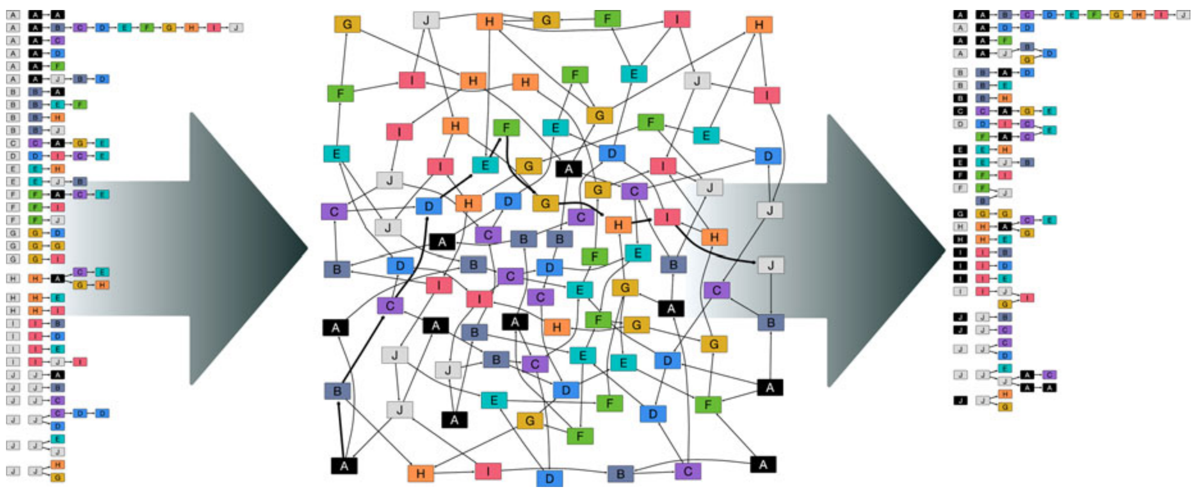


Fig. 3. Example Small Generated Ontology. Rules on the left are used to generate the graph of 100 nodes and the induction method is used to establish the rules on the right. Note that in this example, the induction method produces a grammar with 25% fewer rules by combining some rules together, but also retains a particularly specific rule which demonstrates a linear path between nodes of each type in the graph.

medium ($n = 10{,}000$), and large ($n = 1{,}000{,}000$) numbers of nodes and average degree per node connection degrees of 2, 4, and randomized degree, emulating text, visual design, and ontological design representations. These evaluation sizes were chosen to emulate problems of vastly different levels of complexity. The number of unique nodes in each case was 10% of the total number of nodes in that trial. Trial data objects consisted of strictly typed UUIDs (Universally Unique Identifiers) (Leach et al., 2005) for high-speed comparison.

The implementation was in Python 3 and interconnections were managed with the NetworkX library (Hagberg et al., 2008). All trials were run locally on commodity hardware with eight cores and in instances in which processing lasted longer than 1 h, trials were cut short.

When evaluating performance on synthesized ontologies, the nodes and connections are intrinsic to the representation of the ontology so the first step of the procedure is not explicitly performed and the second step can be conducted directly. Other steps for this evaluation were functionally similar to the text example given in Table 1, with the exception that random starting positions were used, and traversing the graph was done by selecting edges at random from the current operational node. As a consequence, the heavily connected nature of these graphs lead to larger graphs exceeding the allotted computational time and, in some cases, being removed from the evaluation.

Inducing Palladio's Villa

A 3D model of the Villa Foscari, La Malcontenta was produced from reference drawings (Rowe, 1977) using a CAD package, and the load bearing walls and high-level plan were extracted, as shown in Figure 4. These data were chosen, instead of, for example, the entire 3D model, because it more closely reflects the existing grammars that have been created for the dataset facilitating direct comparison with the previously conducted methods. The CAD data were converted into an ontological form in which the points making up each line segment were treated as nodes in the dataset and the lines were interpreted as their connections in the ontology. In other words, only places where lines end or join become critical. For example, a wall is interpreted as just the connection between the start and finishing points of it. Similarly, a door is just a relationship of points where the walls of two rooms meet. This way the grammar making process is not constrained by the physical structure of the building but by the relationships of each point to each other point. Spatial information was retained but not made available to the grammar induction algorithm so that the evaluation would be ontological while building shapes could be visualized in the way shown in Table 1. In this sense, the building data were converted into a graph of points found at junctions of edges from the blueprints, allowing step 1 of the induction method to be conducted in a very straightforward fashion. In steps 2 and 3, feature frequency was used as the primary indicator about rule likelihood, or chunk likelihood, leading to the most common feature patterns being used as fundamental rules. This lead to coherent rules as were shown in Table 1, and meant that expanding rules and establishing interrule relationships could happen without unusual wall segments or stray edges appearing out of place.

Because of the relatively small size of the grammar for the test data, the ontology could be completely parsed to find all rules existing in the data. Computations were performed on commodity computing hardware and all finished in <1 h, so no measures were necessary to accommodate for computational complexity in this part of the study.

Assessment tasks were to attempt to formulate building primitives and to recreate the input data. Palladio's buildings have been the subject of a large amount of research on grammars so rule sets for comparison are readily available, such as from Stiny and Mitchell (1978), and were used as plausible alternatives in the study of the variability of the produced grammars. Conducting this pairwise rule comparison offers insight into conciseness and variability of a grammar.

Inducing piecewise models

The greenhouse model (Fig. 5) was produced using a CAD package and processed using the COLLADA 1.5.0 standard (Barnes & Finch, 2008). The model contained approximately 700 components many of which were instances of other parts but configured in different relationships. All structural and machine components were included in the model but sub-assemblies were ignored when inducing the grammar because they were assumed to be readily purchased in that state (e.g., a pump or a lightbulb would not be further decomposed).

Because of the nature of the COLLADA format, the model had an inherent structured representation so physical proximity was used to transcend structural isolation. As a result, step 1 in the induction process involved traversing the COLLADA file and establishing a graph representation based on which components were in physical contact or orientation with one another. In this way, when processing rule chunking (i.e., steps 2 and 3), only physical proximity was considered for building ontological relationships, the hierarchical connection was not. This avoids giving weight to any assumptions made by the model's author about other hierarchical aspects of the design. Similarly, in performing step 4 of the grammar process, establishing interrule relationships, this interpretation affords succinct analysis of the overarching relationships found in the model in a way that a hierarchical representation would hinder.

Results

Results are reported on trial runs of generated datasets for specific complexity levels in Table 2. Reported values are the percent of resulting grammars that express accuracy, variability, and repeatability while reported values for conciseness are the percentage of the ratio of rules to total data-points, which agrees with computational bounds. All reported values are on 20 runs of the induction algorithm on the respective datasets, based on the metrics defined in the section "Related work" of this paper. In data-limited cases, such as the Palladian Villa and the Piecewise model, cross-validation performed by inducing unique subsets of the data was conducted when needed, for each run. A necessary condition of any grammar to be selected was that it would achieve accuracy and variability; all evaluations reported achieved those goals.

Repeatability varied over the course of experimental cycles (Table 2). In the trial data cases, it varied proportionally to connectedness with higher dimensionality leading to significantly fewer repeatable grammars. This is likely because selecting for concise grammars may sometimes lead to certain grammar features being represented more than once. It can be very difficult to determine the difference between two grammars for the same dataset. For the random degree case this is not a problem for small examples, but for larger cases, this might mean that it

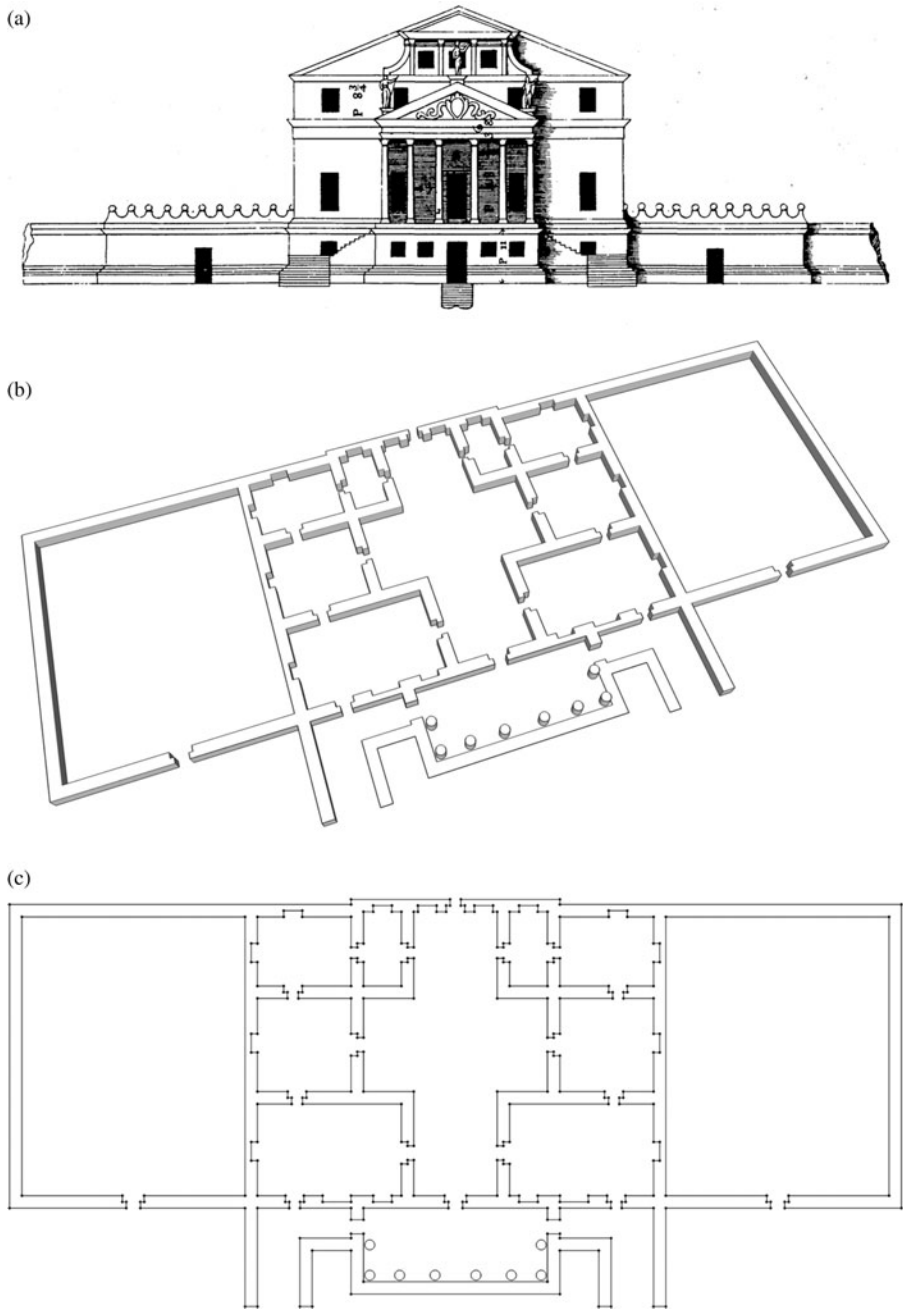


Fig. 4. Villa Malcontenta. (a) Illustration from plate XLIII in Quattro Libri (Rowe, 1977), (b) CAD rendition of features extracted from plan drawings of the villa. (c) Visualization of the underlying graph with node position from coordinates in CAD information.

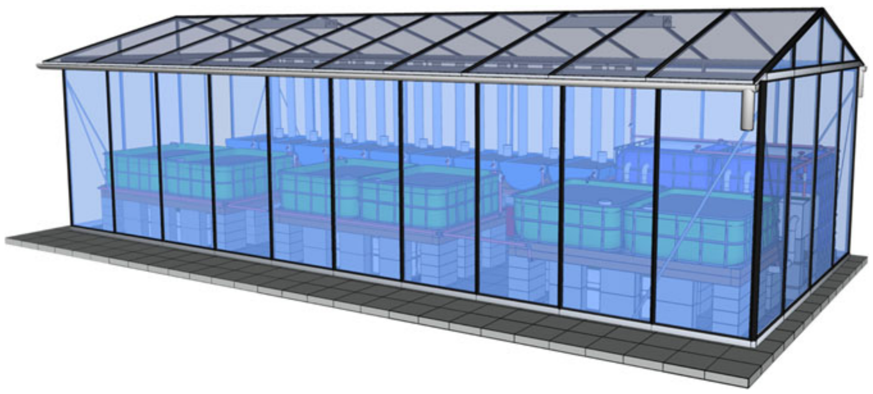


Fig. 5. Greenhouse model. A CAD model of an aquaponics greenhouse with automated delivery systems [Snowgoose (2014). Aquaponics system 8 × 3.4, 3D Warehouse.]. Components down to the bricks and fasteners are individually modeled and are represented discretely in a hierarchy. The entire model was induced.

Table 2. Grammar metrics

Trial	Accuracy (% achieved)	Variability (% achieved)	Repeatability (% achieved)	Conciseness (% rules/*n*)
Degree 2 Ontology	100%±0	100%±0	90%±3	52%±1
Degree 4 Ontology	100%±0	100%±0	80%±4	59%±4
Random Degree Ontology	100%±0	100%±0	78%±9	61%±4
Palladio's Villa	100%±0	100%±0	87%±2	56%±3
Piecewise model	100%±0	100%±0	74%±5	51%±1

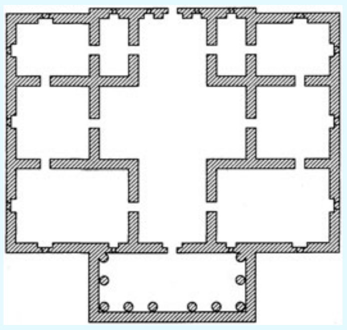
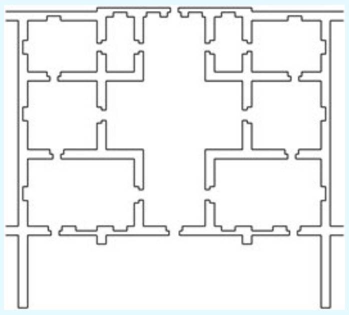
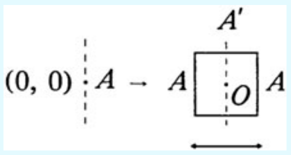
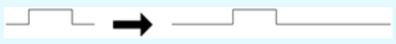
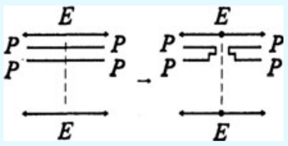
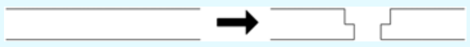
Accuracy and variability are necessary conditions. Repeatability is desired at 100% and conciseness, reported as a percentage ratio of induced rules to unique nodes, is dependent on the type of data being induced. Lower bounds are reported, indicating the first quantile of runs. Calculations are based on the specific approaches introduced in Section "Related Work" of this paper. Evaluations with runtime over 1 h were omitted. In instances providing only one dataset to evaluate, such as the Palladian Villa and the Piecewise model, cross-validation was used, inducing two unique subsets of the data for each comparator. For repeatability, comparisons are conducted between induced grammars from either cross-validation or sample diversity.

becomes almost impossible to rely on only searching for unique rules over all resulting grammars when the algorithm is run.

Conciseness performed best on 1D cases but showed significantly less favorable results in more complex evaluations. This is attributable to the number of a plausible and actual node connected pairs increasing significantly with more dimensionality. As a result, however, it means that the inducted grammars are too large to be useful and are likely to include many rules that are near duplicates of each other. Removing these is non-trivial and would make the current approach orders of magnitude longer to compute. This said efficient rule reduction occurred in many cases. This is demonstrated by the example shown earlier in Figure 2 of a random degree ontology evaluation in which the induced grammar is shorter and simpler than the input grammar while retaining key high complexity rules, and adding multi-function rule alternatives.

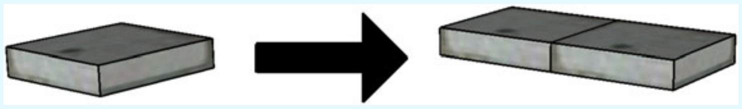
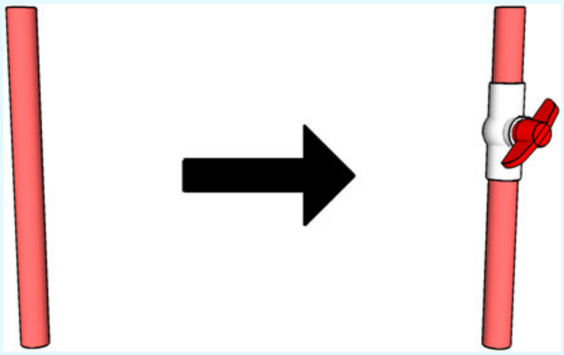
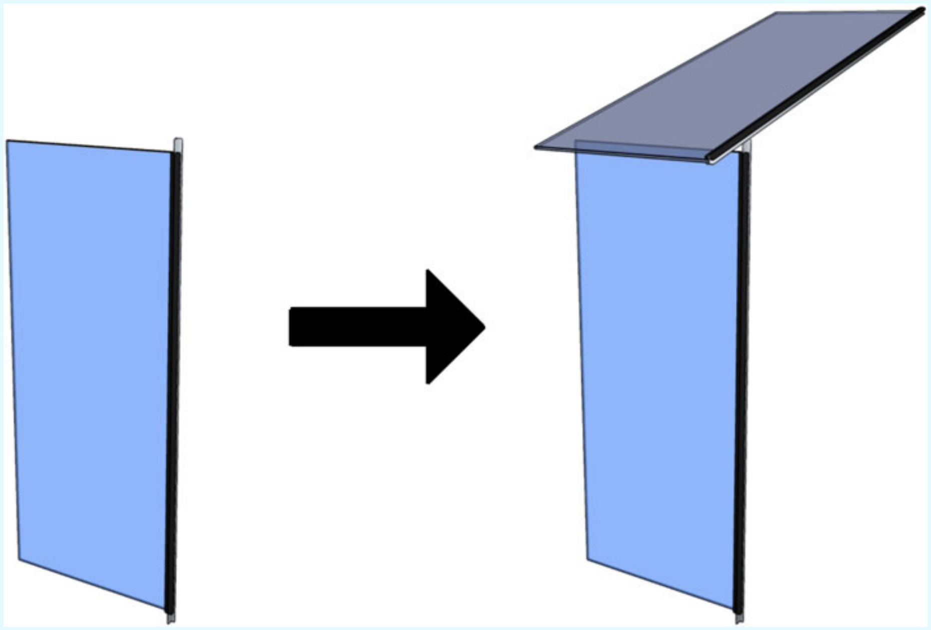
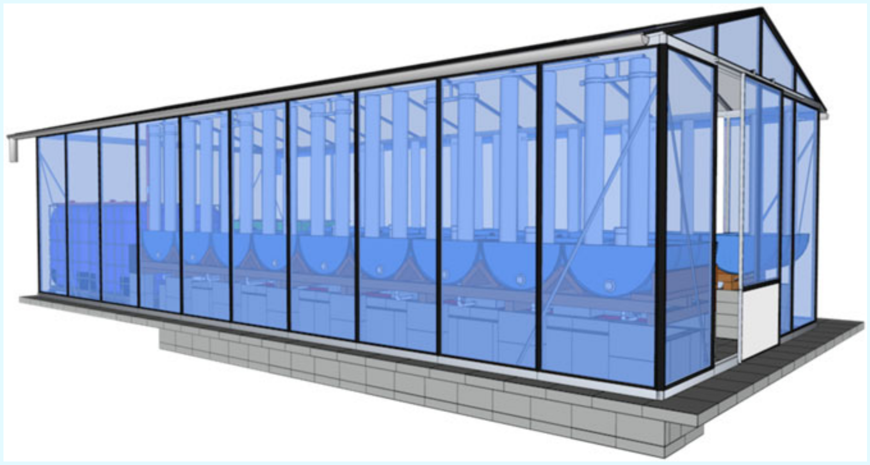
The Palladian Villa grammar demonstrated similar results to the other evaluations. The induced grammar is able to almost perfectly express its source data and fail only where aspects of the source data were not fully induced. For example, the columns at the front of the house as well as the entryway stairs were not properly resolved because rules for those features had no ontological connection to the rest of the structure. This is not the case when the grammar is created using positional data but as

Table 3. Palladian grammar comparison

Data type	Original Palladian Villa (Stiny & Mitchell, 1978)	Automatically induced
Output		
First rule		
Door rules		

A comparison of grammar rules and results from foundational work on the Villa Malcontenta (Stiny & Mitchell, 1978) and an automatically induced grammar from the same source data. The outputs are largely identical but exhibit subtle differences relating to connected elements and the rendering of detail. For instance, more detail is retained in the automated output however that is due to how the manual approach was conducted, not because of fundamental differences in the approaches. The first rules are very different. Where the original approach started with grid definition, the automated approach chooses an arbitrary start point. Door rules show similarity; however, symmetry is not explicit in the automated approach. Rather, it is inferred by the ontology because components found to be part of the door making process cause the symmetry of the rule to emerge. Many rules are not directly comparable because of the difference in approaches but the results indicate that both can produce similar final results.

Table 4. Piecewise ontology induction

Highly general rule	
Highly specific rule	
Traditional structural rule	
Example output	

The highly general rule shows the combining of bricks as one of the rules in forming the base structure. In this model, there are several ways bricks relate but by using chunked rules only rules about direct relationships are needed and these can be combined with higher level rules about the layout of the system to achieve the complete base. On the other hand, highly specific rules such as the one showing adding a valve to a segment of pipe are used to position parts in their context to subassemblies, such as the valve, or such as a vat next to which this assembly would appear. A more traditional structural rule is also included showing a wall segment being used to position a roof segment. The example output shows a different configuration of vats with the associated changes in floor layout and pipe layout.

previously mentioned, the intention was to build an ontological grammar from this data, because it is significantly more challenging to build grammars with no structural coding.

In this process, key differences between our automated grammar induction technique and the traditional human grammar generation methods become apparent as shown in Table 3. For instance, the rule system introduced by Stiny and Mitchell (1978), is largely top-down, moving from the largest layout and symmetry rules to smaller details later in the ruleset which is seemingly built to augment the previously constructed forms. This approach is visible in others' assessments of the same buildings (Benrós et al., 2012); however, it is not the only plausible approach as Stiny and Mitchell recognize. Using a bottom-up approach, as is the case with our automated tool, derives high-level constraints in the context of the details that relate to them. In this way, there are some differences in the strictness with which a design might be determined to be part of the same style. With the bottom-up approach, grammar outputs always exhibit both plan-level stylistic features, and detail design-level stylistic features, for instance, large-scale symmetry as relationships between doors and windows in the same room. On the other hand, the top-down approach develops grammar rules that can be applied to generate buildings that are suitably symmetrical but have none of the low-level features of the style.

The piecewise model was also induced successfully, and this example provides a context for demonstrating how efficient a grammar can become when it is induced over a purely ontological model of a dataset, even if those data are 3D in nature. For example, rules extracted for this approach saw either very high repeated use or they implemented unique features. This shows that the induction process was able to build representative chunks that encompassed much of the inherent variability of details of the model. The rule set for such a complex model is rather large, encompassing over 800 rules governing 3900 components, so only a small number of example rules are shown in Table 4, with the addition of an example alternative output from the induced ruleset. This example offers a striking case for using the shape agnostic, ontological approach adopted in this work. Parts in the induced model exhibited positional patterns but also functional patterns which emerged from broader symmetry and positional relationships. For example, the positioning of pipes relative to valves and vats was performed accurately with the induced grammar in the example output in Table 4, even though these components had different physical relationships in the original induced model.

Conclusions

This work has introduced a method of efficiently inducing shape and graph grammars of complex design contexts using un-coded data. The method was validated with experiments establishing the accuracy, variability, repeatability, and conciseness of produced grammars, with several examples including abstract data and design data. Speed and accuracy challenges are reduced but remain somewhat critical, so these will be an important part of our future work. Fast and accurate machine translation remains a focus of many researchers in computer science and linguistics; incorporating more lessons from their approaches, as well as considering other ways in which automated grammar approaches can be developed and applied are directions of deep interest to advance this work.

The difficulty of computing grammars used for design has made them relatively uncommon in the industry. Facilitating their efficient use with this new automated approach may rekindle the formal assessment of shape in design outside of the academic community in computer-aided design software, commercial design methods, and other fields that utilize special information. Furthermore, the academic advantages of having fast ways to generate grammars for a broad array of types of data open the field to new grammar-based endeavors.

Acknowledgments. This work was funded by the National Science Foundation under grants CMMI-1160840 and CMMI-1233864. This paper extends earlier work published in *Design Computing and Cognition 2016* in Evanston, Illinois (Whiting, Cagan and Leduc 2016).

References

Ates K and Zhang K (2007) Constructing VEGGIE: machine learning for context-sensitive graph grammars. In *Proceedings – International Conference on Tools with Artificial Intelligence, ICTAI*, pp. 456–463.

Babai L (2015) Graph Isomorphism in Quasipolynomial Time. *arXiv* 7443327, 84.

Babai L, Kantor WM and Luks EM (1983) Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*, pp. 162–171.

Balahur A and Turchi M (2014) Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language* **28**(1), 56–75.

Barnes M and Finch EL (2008) *Collada-Digital Asset Schema Release 1.5.0, Specification*. Clearlake Park, CA: Khronos Group.

Benrós D, Hanna S and Duarte JP (2012) A generic shape grammar for the Palladian Villa, Malagueira house, and prairie house. *Design Computing and Cognition '12'* **12**(18), 321–340.

Berwick RC and Pilato S (1987) Learning syntax by automata induction. *Machine Learning* **2**(1), 9–38.

Chau HH, Chen X, McKay A, and de Pennington A (2004) Evaluation of a 3D shape grammar implementation. In Gero JS (ed.). *Design Computing and Cognition '04*. Dordrecht: Springer.

DeNero J and Uszkoreit J (2011) Inducing sentence structure from parallel corpora for reordering. In *EMNLP 2011 – Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 193–203.

Ding Y and Palmer M (2005) Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, vol **38** (June), pp. 541–48.

Fouss F, Pirotte A, Renders JM and Saerens M (2007) Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering* **19**(3), 355–69.

Gero JS (1994) Towards a model of exploration in computer-aided design. In Gero JS and Tyugu E (eds). *Formal Design Methods for Computer-Aided Design*. Amsterdam: North-Holland, pp. 315–336.

Gips J (1999) Computer implementation of shape grammars. In *Proc. Workshop on Shape Computation, MIT*. Accessed at http://www.shapegrammar.org/implement.pdf

Hagberg AA, Schult DA and Swart PJ (2008) Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pp. 11–15.

Kang U, Tong H and Sun J (2012) Fast random walk graph kernel. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 828–838.

Knuth DE (1998) *The Art of Computer Programming Volume 3. Sorting and Searching*. Reading, MA: Addison Wesley, vol. **3**, p. 829.

Königseder C and Shea K (2015) Analyzing generative design grammars. In *Design Computing and Cognition '14*, pp. 363–381.

Kudo T and Matsumoto Y (2002) Japanese dependency analysis using cascaded chunking. In *Proceeding of the 6th Conference on Natural language learning – COLING-02*, vol. 20, pp. 1–7.

Leach P, Mealling M and Salz R (2005) *A Universally Unique IDentifier (UUID) URN Namespace*. The Internet Society, pp. 1–32.

Lee YS and Wu YC (2007) A robust multilingual portable phrase chunking system. *Expert Systems with Applications* **33**(3), 590–599.

McCormack JP and Cagan J (2002) Designing inner hood panels through a shape grammar based framework. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing* **16**(4), 273–290.

McKay BD and Piperno A (2014) Practical graph isomorphism, II. *Journal of Symbolic Computation* **60**, 94–112.

Mikolov T, Le QV and Sutskever I (2013) Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168v1*, 1–10.

Orsborn S and Cagan J (2009) Multiagent shape grammar implementation: automatically generating form concepts according to a preference function. *Journal of Mechanical Design* **131**(12), 121007.

Piazzalunga U and Fitzhorn P (1998) Note on a three-dimensional shape grammar interpreter. *Environment and Planning B: Planning and Design* **25**(1), 11–30.

Rawson K and Stahovich TF (2009) Learning design rules with explicit termination conditions to enable efficient automated design. *Journal of Mechanical Design, Transactions of the ASME* **131**(3), 031011-031011–11.

Rowe C (1977) Mathematics of the ideal villa and other essays. *Jae* **31**, 48.

Rozenberg G (1997) Handbook of graph grammars and computing by graph transformation. *Handbook of Graph Grammars* **1**, 1–8.

Sánchez-Martínez F and Pérez-Ortiz JA (2010) Philipp Koehn, statistical machine translation. *Machine Translation* **24**, 273–278.

Schmidt LC and Cagan J (1997) GGREADA: a graph grammar-based machine design algorithm. *Research in Engineering Design* **9**(4), 195–213.

Schnier T and Gero JS (1996) Learning genetic representations as alternative to hand-coded shape grammars. In *Artificial Intelligence in Design '96*. Dordrecht: Springer, pp. 39–57.

Schwenk H (2012) Continuous space translation models for phrase-based statistical machine translation. *COLING (Posters)* (December), pp. 1071–1080.

Slisenko AO (1982) Context-free grammars as a tool for describing polynomial-time subclasses of hard problems. *Information Processing Letters* **14**(2), 52–56.

Speller TH, Whitney D and Crawley E (2007) Using shape grammar to derive cellular automata rule patterns. *Complex Systems* **17**(1/2), 79–102.

Stiny G (1980) Introduction to shape and shape grammars. *Environment and Planning B* **7**(3), 343–351.

Stiny G and Mitchell WJ (1978). The palladian grammar. *Environment and planning B: Planning and Design* **5**(1), 5–18.

Stolcke A and Omohundro S (1994) Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications*, pp. 106–118.

Suh NP (2001) *Axiomatic Design: Advances and Applications*. New York: Oxford University Press.

Talton J, Yang L, Kumar R, Lim M, Goodman N and Měch R (2012) Learning design patterns with Bayesian grammar induction. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology – UIST '12*, p. 63.

Trescak T, Esteva M and Rodriguez I (2012) A shape grammar interpreter for rectilinear forms. *CAD Computer Aided Design* **44**(7), 657–670.

Trescak T, Rodriguez I and Esteva M (2009) General shape grammar interpreter for intelligent designs generations. In *Proceedings of the 2009 6th International Conference on Computer Graphics, Imaging and Visualization: New Advances and Trends, CGIV2009*, pp. 235–240.

Yue K and Krishnamurti R (2013) Tractable shape grammars. *Environment and Planning B: Planning and Design* **40**(4), 576–594.

Mark E. Whiting is a PhD candidate co-advised by Jonathan Cagan and Philip Leduc. He has a Bachelor of Design from RMIT in Australia, and a Master's of Science from KAIST in South Korea, both focusing on Industrial Design. Mark has worked in design and innovation strategy in a consulting capacity around the world, and as a researcher with groups at KAIST and Stanford.

Jonathan Cagan is the George Tallman and Florence Barrett Ladd Professor in Engineering, Department of Mechanical Engineering, at Carnegie Mellon University, with an appointment in the School of Design. Cagan serves as Associate Dean for Strategic Initiatives for the College of Engineering at CMU, in part leading the strategy for its Silicon Valley campus. He co-founded and co-directs the Integrated Innovation Institute and Master of Integrated Innovation for Products and Services, and serves as the Head of CMU's MS in Software Management and MS in Technology Ventures programs. Cagan is also Faculty Co-Director for the Swartz Center for Entrepreneurship.

Philip LeDuc is the William J. Brown Professor in the Department of Mechanical Engineering at Carnegie Mellon University with appointments in Biomedical Engineering, Computational Biology, and Biological Sciences. He received his PhD from Johns Hopkins University and did a fellowship at Children's Hospital and Harvard Medical School. He has received the National Science Foundation CAREER Award, the Beckman Foundation Young Investigators Award, and been selected as a faculty member for the Sloan Foundation minority PhD Program. *PNAS* and *Nature* have featured his work. He has filed numerous patents, has started companies, and has consulted for a diversity of companies.