Grammar Inference Technology Applications in Software Engineering

Barrett R. Bryant[1], Marjan Mernik[1,2], Dejan Hrnčič[2], Faizan Javed[3], Qichao Liu[1], and Alan Sprague[1]

[1] The University of Alabama at Birmingham, Department of Computer and Information Sciences, Birmingham, AL 35294-1170, U.S.A.
{bryant,mernik,qichao,sprague}@cis.uab.edu
[2] University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Smetanova 17, SI-2000 Maribor, Slovenia
{marjan.mernik,dejan.hrncic}@uni-mb.si
[3] Regions Financial Corp., Mortgage Shared Systems, Birmingham, AL 35244, U.S.A.
faizan.javed@regions.com

Abstract. While Grammar Inference (GI) has been successfully applied to many diverse domains such as speech recognition and robotics, its application to software engineering has been limited, despite wide use of context-free grammars in software systems. This paper reports current developments and future directions in the applicability of GI to software engineering, where GI is seen to offer innovative solutions to the problems of inference of domain-specific language (DSL) specifications from example DSL programs and recovery of metamodels from instance models.

Keywords: domain-specific languages, grammar inference, metamodel.

1 Introduction

Grammatical inference (GI), often called grammar induction or language learning, is the process of learning a grammar from positive and/or negative sentence examples. Machine learning of grammars finds many applications in syntactic pattern recognition, computational biology, and natural language acquisition [3]. In the realm of software engineering, context-free grammars (CFGs) are of paramount importance for defining the syntactic component of programming languages. Grammars are increasingly being used in various software development scenarios, and recent efforts seek to carve out an engineering discipline for grammars and grammar dependent software [6].

The structure of this paper is as follows. Section 2 explores the application of GI to inferring domain-specific language (DSL) specifications from example DSL programs. In Section 3, application of GI to recovery of a metamodel from instance models is discussed. We conclude in Section 4.

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 276–279, 2010.

2 Inferring DSL Specifications from DSL Programs

A domain-specific language (DSL) is a programming language dedicated to a particular domain or problem [9]. It provides appropriate built-in abstractions and notations; it is usually small, more declarative than imperative, and less expressive than a general-purpose language (GPL). Using grammar inference, language specifications can be generated for DSLs, facilitating programming language development for domain experts not well versed in programming language design. This enables a domain expert to create a DSL by supplying sentences from the desired DSL to the grammar induction system, which would then create a parser for the DSL represented by those samples, thus expediting programming language development.

A memetic algorithm [10] is a population-based evolutionary algorithm with local search. MAGIc, **M**emetic **A**lgorithm for **G**rammatical **I**nferen**c**e [8] infers CFGs from positive samples, which are divided into a learning set and a test set. The learning set is used only in local search, while grammar fitness is calculated on samples from the learning and test sets. Using samples from the test set in the grammar inference process is the main difference between our approach and many machine learning approaches, where the test set is used for testing the result accuracy. Although the initial population has been created mostly randomly in evolutionary algorithms such an approach has been proven insufficient for grammar inference [1]. Indeed, a more sophisticated approach is needed and an initial population of grammars is generated using the Sequitur algorithm [11], which generates a grammar that only parses a particular sample from a learning set. Hence, Sequitur does not generalize productions. Moreover, the initialization procedure can be enhanced with seeds of partially correct grammars or grammar dialects, which are useful for learning grammar versions [2].

We have tested MAGIc on the formal language $\{a^n b^n c^m | n, m \geq 1\}$ used in [13] and also on various DSLs, including desk calculator language DESK [12] and a DSL for describing 3-D shapes and textures [15]. MAGIc is able to infer CFGs, which are non-ambiguous and of type LR(1). Often these have fewer productions than the original grammar. Further details may be found in [8].

3 Recovery of a Metamodel from Instance Models

GI can also be applied to solve problems in domain-specific modeling (DSM) [14]. DSM assists subject matter experts in describing the essential characteristics of a problem in their domain. The typical approach to DSM involves the construction of a metamodel, from which instances are defined that represent specific configurations of the metamodel entities. Throughout the evolution of a metamodel, repositories of instance models (or domain models) can become orphaned from their defining metamodel. In such a case, instance models that contain important design knowledge cannot be loaded into the modeling tool due to the version changes that have occurred to the metamodel. Thus, the ability to recover the design knowledge in a repository of legacy models is needed. A correspondence exists

between the domain models that can be instantiated from a metamodel, and the set of programs that can be described by a grammar. This correspondence can be leveraged by creating a GI-based tool that allows the recovery of lost metamodels by making use of information contained in the domain models.

MARS (MetAmodel Recovery System) [4][7] is a system we have developed to infer metamodels from model instances, based on the GME[1] modeling tool. GME, and most other modeling tools, represent the model instance using XML. XSLT[2] is a visual-to-textual representation transformation process which we use for reading in a set of instance models in XML and translating them into a DSL, called the Model Representation Language (MRL), that accurately represents the visual GME domain models contained in XML files. MRL is then loaded into the LISA language description environment [9]. As a result of the inference process, an inferred metamodel in XML is generated using formal transformation rules. The inferred metamodel can then be used by GME to load the previous instance models into the modeling tool.

MARS has been validated using example models developed in ESML (Embedded Systems Modeling Language), a domain-specific graphical modeling language for modeling embedded systems [5]. We created various model instances exhibiting different properties and were able to infer a very good approximation of the metamodel. The quality of the inferred metamodel depends on the level of details available in the domain instances and also the number of domain instances used. If the set of domain instances used in the inference did not make use of all the constituent elements of the original metamodel, then those elements cannot be recovered in the inferred metamodel. Almost all properties addressed in the instances we developed were used properly in the metamodel inference and the cardinalities of connections appeared in the original were also correctly computed. Further details may be found in [7].

4 Conclusions

We have shown that grammar inference may be used to infer DSLs from example DSL source programs and metamodels from model instances. In both cases results are good when the sample set is sufficient to illustrate the key aspects of the underlying grammar. There are still some limitations in inferring highly complex recursive structures. We are currently working on improving our algorithms to overcome this. It is also the case that for both DSLs and models, the addition of semantic information would be beneficial. Such semantic information could allow the extension of the construction of DSL parsers into complete DSL compilers, and also allow for more refined metamodels to be inferred using semantic information.

Acknowledgments

This work was supported in part by United States National Science Foundation award CCF-0811630.

[1] Generic Modeling Environment - http://www.isis.vanderbilt.edu
[2] Extensible Stylesheet Transformation - http://www.w3.org/TR/xslt

References

1. Črepinšek, M., Mernik, M., Javed, F., Bryant, B.R., Sprague, A.: Extracting grammar from programs: Evolutionary approach. ACM SIGPLAN Notices 40(4), 39–46 (2005)
2. Dubey, A., Jalote, P., Aggarwal, S.K.: Learning context-free grammar rules from a set of program. IET Software 2(3), 223–240 (2008)
3. de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press, Cambridge (2010)
4. Javed, F., Mernik, M., Gray, J., Bryant, B.: MARS: A metamodel recovery system using grammar inference. Information and Software Technology 50, 948–968 (2008)
5. Karsai, G., Neema, S., Sharp, D.: Model-driven architecture for embedded software: A synopsis and an example. Science of Computer Programming 73(1), 26–38 (2008)
6. Klint, P., Lämmel, R., Verhoef, C.: Toward an engineering discipline for grammarware. ACM Transactions on Software Engineering Methodology 14(3), 331–380 (2005)
7. Liu, Q., Bryant, B.R., Mernik, M.: Metamodel recovery from multi-tiered domains using extended MARS. In: Proc. COMPSAC 2010, 34th Annual International Computer Software and Applications Conference (to appear, 2010)
8. Mernik, M., Hrnčič, D., Bryant, B.R., Javed, F.: Applications of grammatical inference in software engineering: Domain specific language development. In: Martin-Vide, C. (ed.) Scientific Applications of Language Methods, pp. 475–511. Imperial College Press (2010)
9. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys 37(4), 316–344 (2005)
10. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. rep., California Institute of Technology, Concurrent Computation Program 158-79 (1989)
11. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: a linear-time algorithm. Journal of Artificial Intelligence Research 7(1), 67–82 (1997)
12. Paakki, J.: Attribute grammar paradigms—a high-level methodology in language implementation. ACM Computing Surveys 27(2), 196–255 (1995)
13. Sakakibara, Y.: Learning context-free grammars using tabular representations. Pattern Recognition 38(9), 1372–1383 (2005)
14. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. IEEE Computer 39(2), 25–31 (2006)
15. Strnad, D., Guid, N.: Modeling trees with hypertextures. Computer Graphics Forum 23(2), 173–187 (2004)