

Can Graph Grammars Make Formal Methods More Human?

MAURO PEZZÈ

Politecnico di Milano, Italy

LUCIANO BARESI

Politecnico di Milano, Italy

Abstract

Formal methods are scarcely used in industrial applications. Industrial strength tools and educational effort do not significantly help in promoting formal methods. Main obstacles to the industrial application of formal methods are lack of flexibility and specialization, and difficulties in interpreting analysis results. This paper argues that graph grammars can help in overcoming such limitations and suggests translation rules, based on graph grammars, as a natural merging of formalizations based on graph grammars and rule based approaches, thus indicating a challenging practical application domain for graph grammars.

Keywords

Formal Methods, Software Specification, Graph Grammars, Rule Based Translation.

1 Introduction

Despite the enormous amount of investments in research and development, industrial applications of formal methods are negligible [8, 20]. Industrial strength tools produced in the last decade ([10, 16]) and educational effort of many universities suggest that lack of automation and skills are not the key obstacles to the use of formal methods, as often claimed. The real obstacles in applying formal methods in industry can be found elsewhere: lack of flexibility, lack of specialization, difficulties in expressing properties and examining analysis results [14].

Formal methods are extremely rigid. Anyone who has been working with formal methods beyond simple toy examples has faced the problem of capturing specific elements of a description with a chosen formal method, regardless the method and the application domain. Most of us know the frustration of spending a conceivable amount of time to find a representation of a fairly simple construct in an unsuitable formalism. Formal methods are general purpose: Petri nets, finite

state automata, temporal logics, just to cite few examples, apply to almost every domain. Very few formal methods, if any, are tailored to a specific domain. Although generality can be a precious quality, complete absence of specialization can result in difficulties in meeting all aspects of the specific domains, as often experienced in industrial settings [8, 14]. Interesting properties cannot always be expressed easily and results are often difficult to be decoded and related back to the original formulation of the problem.

The recently proposed solutions to the aforementioned problems have not been successful so far. This paper suggests that a rule based framework founded on graph grammars can lead to an acceptable solution to these problems and indicates some challenges that need to be addressed to open formal methods to a large set of industrial applications.

2 Gateways to Formal Methods

Lack of flexibility and difficulties in expressing properties and analysis results can be overcome by filtering the access to the chosen formal method. Lotos ([6]) and SDL([19]) are examples of how suitable domain specific notations can help in accessing formal methods. Lotos *masks* process algebras with constructs tailored to the specific domain and makes properties and results easily accessible to domain experts. Similarly, SDL *masks* finite state automata to ease their use. Both approaches are designed for specifying and analyzing communication protocols. Unfortunately these and few other successes cannot be immediately replicated in other domains, since few application domains present the same stability of communication protocol and thus can be addressed with fixed notations.

Often, specific problems require that notations be suitably tailored or even new notations be defined. These needs have been experienced even with formal methods, as witnessed by the many variants of the most successful models (e.g., Petri nets, temporal logics, process algebras), but are exasperated with informal notations. The most popular informal specification notations, e.g., structured analysis and object oriented design notations, have been used in so many syntactic and semantic variants that we can identify whole notation families.

In this papers, we argue that graph grammars can provide the framework for masking formal methods with flexible user notations, and thus can support a large variety of application domains, including most popular informal specification notations used in software engineering.

A first step towards a solution is represented by the attempts to formally define graphical notations with graph grammars. Examples are the graph grammar supported editors proposed by the Berlin school ([1]) and the three-grammars approach by the Aachen school ([21]). These works indicate the possibility of using graph grammars to interpret user-friendly graphical notations, but do not provide the flexibility required in many application domains.

A complementary step in this direction is represented by the attempts to define rules for mapping informal specifications notations to formal models. Examples of such approaches are the rule based mapping framework presented in [17] and the recent attempts to formalize UML with rules [11]. Rule based approaches solve the flexibility problem, but hardly adapt to continuously evolving notations, and are thus limited to fairly stable notations.

Graph grammar based rules for mapping informal specification notations to formal models can solve the aforementioned problems by suitably merging the two classes of approaches. A multiple graph grammar setting similar to the one proposed by Andy Schuerr in [21, 18] can be used as the formal framework to define mapping rules that can better cope with continuously evolving notations. Differently from other rule based approaches, the formal foundation of graph grammars can simplify the problems derived from the need of quickly changing interpretation, by supporting modifications of subsets of mapping rules.

Continuously changing specification notations require mapping rules to be frequently modified and adapted to new needs, without losing consistency of both the mapping onto the formal model and the interpretation of the analysis results. Changing a mapping rule can affect several other rules and modify the analysis of important properties and the interpretation of the analysis results. To address rapid changes, we need to be able to easily identify the scope of changes due to the modification of a rule both in the mapping onto the formal model and in the interpretation of properties of interest. Both goals can be achieved with a hierarchical mapping of constructs that preserves locality of elements and limits the scope of changes. The images of elements of the informal specification notation must be locally identifiable in the formal model, and the interactions among mappings of different elements must be reduced to the interaction of their interfaces. In this way, changes to the semantics of single elements remain localized to the rules that deal with the elements themselves, and only changes in the element interfaces can affect several elements. Also in this case, the rules to be modified are limited to the interface rules of the affected elements and can easily be identified.

A hierarchical *local* mapping together with the choice of the same language to express properties at the specification notation and the formal model levels can simplify the expression of properties of interest and the presentation of the analysis results. Properties of interest can be parametric with respect to the elements and results of analysis can be mapped back by reverting the local mapping. For example, let us assume that a data transformation at the specification notation level is mapped onto a Petri net that includes a transition *Start* to model the start of the transformation, as shown in Figure 1. The property that every data transformation at the interface notation level can be eventually executed can be expressed with a simple temporal logic formula that predicates on the transformations. This property can easily be mapped onto a temporal logic property with the same structure that predicates on the corresponding semantic elements, i.e., the Petri net transitions of type *Start*. A negative result that indicates a transition of type *Start*, that

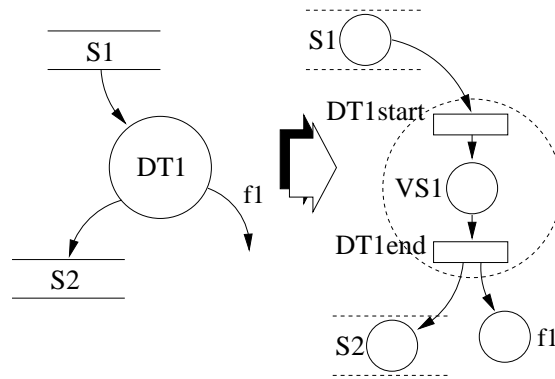


Figure 1: A Petri net giving precise semantics to SA data transformations

cannot be eventually executed can be easily mapped into a comprehensible form by reversing the mapping that associate each functional transformation to a transition of type *Start*. A concrete framework based on a pair of graph grammars to map informal specification notations to Petri nets is presented in [2, 3].

3 Open Problems

Several features can strongly affect the mapping of informal specification notations to formal models, among them: hierarchy, global elements, state, incompleteness and inconsistency, and extensibility of the informal notation. For example, structured analysis real time (SA-RT) notations allow processes to be hierarchically decomposed and allow control to act on subtrees of the hierarchy ([13]); Statecharts allow hierarchically decomposed states to be entered according to a history tag that indicates the last visited sub-state ([12]); UML allows the same feature to be described with different, not always consistent, diagrams, and allows partial descriptions of the system ([7]); FBD, that is, the IEC 1131-3 standard language for modeling programmable controllers, allows new notation elements to be added incrementally to the specification language ([15]). These and other features may complicate the rule based approach and make it impractical in many interesting cases.

Preliminary experiments of mapping SA-RT, Statecharts, UML, FBD, and other domain specific notations indicate that the potential problems identified so far do not represent main obstacles for a graph grammar based approach ([2, 3, 4]). We believe that the possibility of mapping such a large set of complex features is largely due to the strong mathematical foundation of graph grammars, and we doubt that other less formal rule based approaches can achieve the same level of

generality.

The preliminary experiments done so far suggest that the large set of complex graph grammar rules needed to fully mask the details of a formal model may be difficult to manage at industrial scale. The results of using a pre-competitive prototype for FBD to model and analyze sample industrial applications are extremely encouraging. However, further research and experiments are required to simplify the proposed approach and thus reach the equilibrium between formality, capability of changing rules on the fly without affecting the correctness of the translation, and user friendliness.

4 Conclusions

In this paper, we illustrated a possible practical application of graph grammars to provide a flexible framework for formalizing informal specification notations. The proposed framework aims at merging results obtained in previous projects and can represent an interesting application of graph grammars. A similar approach can be applied beyond the area of specification notations. Examples of other application domains where graph grammars are proposed as a possible solution are software processes ([?]), software architectures ([9]), programming languages ([5]).

References

- [1] R. Bardohl, M. Minas, A. Schrr, and G. Taentzer. *Application of Graph Transformation to Visual Languages*, pages 103–180. 1999.
- [2] L. Baresi. *Formal Customization of Graphical Notations*. PhD thesis, Dipartimento di Elettronica e Informazione – Politecnico di Milano, 1997. in Italian.
- [3] L. Baresi, A. Orso, and M. Pezzè. Introducing Formal Methods in Industrial Practice. In *Proceedings of the 20th International Conference on Software Engineering*, pages 56–66. ACM Press, 1997.
- [4] L. Baresi and M. Pezzè. On Formalizing UML with High-Level Petri Nets. Technical Report 09.98, Dipartimento di Elettronica e Informazione – Politecnico di Milano, 1998.
- [5] D. Blostein and A. Schuerr. Computing with Graphs and Graph Transformations. *SOFTPREX: Software–Practice and Experience*, 29, 1999.
- [6] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. In P. H. J. van Eijk, C. A. Vissers, and M. Diaz, editors, *The Formal Description Technique LOTOS*, pages 23–73. Elsevier Science Publishers North-Holland, 1989.

- [7] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. The Addison-Wesley Object Technology Series, 1998.
- [8] E. Clarke and J. Wing. Formal Methods: State of the Art and Future Directions. Technical report, ACM, August 1996. Strategic Directions in Computing Research: Formal Methods Working Group (Group Report).
- [9] D. Le Métayer. Describing Software Architecture Styles using Graph Grammars. *IEEE Transactions on Software Engineering*, 24(7):521–533, 1998.
- [10] P. G. Larsen R. Elmstrøm and P. B. Lassen. The IFAD VDM-SL Toolbox: A Practical Approach to Formal Specifications. *ACM Sigplan Notices*, 29(9):77–80, September 1994.
- [11] Robert France, Andy Evans, Kevin Lano, and Bernhard Rumpe. The UML as a formal modeling notation. *Computer Standards & Interfaces*, 19(7):325–334, November 1998.
- [12] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, (8), 1987.
- [13] D. J. Hatley and I. A. Pirbhai. *Strategies for Real-Time System Specification*. Dorset House, New York, 1987.
- [14] C. Heitmeyer. On the Need for “Practical” Formal Methods. 1486:18–25, 1998.
- [15] IEC. Part 3: Programming Languages, IEC 1131-3. Technical report, International Electrotechnical Commission - Geneva, 1993.
- [16] S. Owre, J. Rushby, N. Shankar, and F. Von Henke. Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS. *IEEE Transactions on Software Engineering*, 21(2), February 1995.
- [17] R.F. Paige. A meta-method for formal method integration. *Lecture Notes in Computer Science*, 1313:473–485, 1997.
- [18] J. Rekers and A. Schürr. A Graph Based Framework for the Implementation of Visual Environments. In *Proceedings of VL'96 12th International IEEE Symposium on Visual Languages*. IEEE-CS Press, September 1996.
- [19] O. Færgemand and A. Olsen. Introduction to SDL-92. *Computer Networks and ISDN Systems*, 26:1143–1167, 1994.
- [20] H. Saiedian. An Invitation to Formal Methods. *IEEE Computer*, pages 16–30, April 1996.

- [21] A. Schürr. Specification of Graph Transformations with Triple Graph Grammars. In G. Tinhofer, editor, *Proceedings WG 94 20th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 903 of *Lecture Notes in Computer Science*, pages 151–163. Springer-Verlag, June 1994.

Mauro Pezzè is with Dipartimento di Elettronica e Informazione at Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy. Phone: +39 02 2399 3523, Fax: +39 02 2399 3411. E-mail: pezze@elet.polimi.it

Luciano Baresi is with Dipartimento di Elettronica e Informazione at Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy. Phone: +39 02 2399 3638, Fax: +39 02 2399 3411. E-mail: baresi@elet.polimi.it