Provenance Meets Bidirectional Transformations

Anthony Anjorin
Department of Computer Science
Paderborn University, Germany
anthony.anjorin@upb.de

James Cheney
LFCS, School of Informatics
University of Edinburgh and The Alan Turing Institute
jcheney@inf.ed.ac.uk

Abstract

Bidirectional transformations (bx) manage consistency between different independently-changing data structures, such as software engineering models. Many bx tools construct, exploit, and maintain various auxiliary structures required for correct and efficient consistency management. These data structures seem analogous to *provenance* in other settings, but their design is often ad hoc and implementation-dependent. However, it is increasingly urgent to rationalize their design and use as first-class *explanations*, to help users understand complex system behavior. In this paper we explore whether and how these auxiliary structures can already be viewed as forms of provenance, and outline open questions and possible future directions for provenance in bidirectional transformations, and vice versa.

Keywords Provenance, Triple Graph Grammars, Bidirectional Transformations, Consistency Management

1 Introduction

The development and maintenance of complex software systems can be made tractable by decomposing systems into multiple abstractions (models), each chosen to be as suitable as possible for a specific group of experts. The price for this separation into multiple concerns is, however, having to address the challenge of *consistency management*, i.e., being able to check for and possibly restore the consistency of the entire system. The *bidirectional transformation (bx)* research community [7] has focused on different consistency management tasks over the years, producing diverse bx approaches and bx tools. A substantial number of such bx tools are based on *Triple Graph Grammars (TGGs)* [15], a rule-based formalism that can be used to automatically derive consistency management operations from a single specification.

TGG tools construct, exploit, and maintain various auxiliary structures for consistency management, which we call *horizontal alignment*. A wide variety of tools in *Model-Driven Engineering (MDE)* context provide similar capabilities, under a variety of different names [8, 11–13]. Possible connections between provenance and bx have been discussed perennially [6], and bx researchers from the database community have often wondered whether these auxiliary structures constitute provenance information, but the exact relationship is not well-understood. In this short paper, therefore, we attempt to provide a high-level overview of the auxiliary structures typically used by TGG tools for consistency management. We claim that these auxiliary structures already could be considered forms of provenance, but have been established in an ad-hoc manner across various TGG tools, and utilized up until now more as a (technical) means to an end, often inaccessible to end users. We argue that providing dedicated and rich support for provenance information can be pivotal for TGG tools, especially for complex consistency management tasks that require convincing *explanations* to guide conflict resolution and user interaction [16].

This paper is intended as a conversation-starter to provoke discussion between the BX and provenance communities, taking advantage of the co-location of BX 2019 and TaPP 2019 workshops. Although we focus on TGGs for concreteness and cannot go into full technical details, many of these observations also apply to bx formalisms more generally. In the interest of accessibility, we include a glossary of possibly-unfamiliar technical terms in the appendix.

2 Running Example

As our running example, we use a simplified version of the *FamiliesToPersons* benchmark for bidirectional transformations [2]. The scenario involves synchronising two sources of information maintained, e.g., by different government agencies. Figure 1 depicts a source and target metamodel describing the types and relations used in each domain: The source metamodel focuses on *families*, grouping named family members in a named family according to their roles in the family. In contrast, the target metamodel does not have the concept of a family and can only keep track of people as named persons. Note, however, that target models are not "views" that can be completely derived from corresponding source models as only the target metamodel maintains dates of birth of people. A pair of source and target model is consistent if and only if there exists a bijection between family members and persons such that every person's name is equal to the family member's name concatenated together with his/her family's name using `", "` as a separator.

TaPP 2019, June 3, 2019, Philadelphia, PA

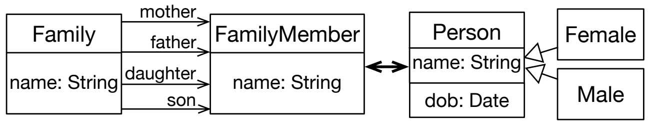

Figure 1. Families (Source) and Persons (Target) Metamodels

This consistency relation can be specified by providing a graph grammar that generates all consistent pairs. With such a grammar, checking if a pair of source and target models is consistent becomes a language membership problem, i.e., can the pair be "parsed" using the rules of the grammar?

As the benchmark does not state that any attribute values are unique identifiers, there can be multiple bijections between a consistent source and target model. To pass all tests in the benchmark, therefore, it is necessary to keep track of which family members correspond to which persons. The TGG approach does this by introducing a third model, the *correspondence* model connecting source and target models. The grammar thus generates a language of consistent *triples* not pairs, and in general, the same source and target models can be present in multiple, different triples in the language. These "connections" between source and target elements is indicated in Fig. 1 by a bold double-headed arrow between `FamilyMember` and `Person`. The exact representation of these connections (typed, attributed, 1-1, many-many) depends on the specific TGG tool [13].

(Triple) graph grammars are a generalisation of string grammars to typed, attributed graphs. Most TGG tools support context-sensitive, monotonic (non-deleting) triple rules without terminal symbols. Figures 2 and 3 depict the triple rules for handling mothers: `CreateMother` (Fig. 2) has no context and creates a family, a family member as a mother in the family, and a corresponding female person when applied. Created elements have a green outline with a ++ markup. The relation between attribute values is specified as a constraint that must be fulfilled when applying the rule.

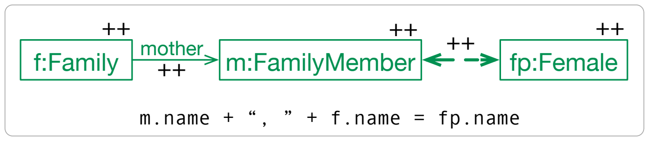

Figure 2. `CreateMother` Triple Rule

`AddMother` (Fig. 3) is similar, but requires an existing family as context, into which the created family member is added as a mother. Context elements have a black outline and no markup. Both rules also establish a correspondence between the newly created mother and the corresponding female person. Exactly how this is represented (a single edge, a node with two edges) depends on the specific TGG tool. The TGG for our running example comprises similar rules for handling fathers, daughters, and sons. In addition to the required bijection, note that triples with empty families, or with family members that are not in a family, are inconsistent according to the TGG as they can never be generated.

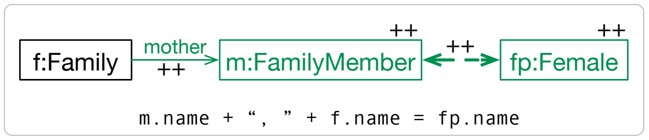

Figure 3. `AddMother` Triple Rule

Similar to parser generators, TGG tools can derive different consistency management operations from a TGG, including *model generation* to generate large examples for scalability and conformance testing; *model transformation* to construct a target consistent with a given source (or vice versa); *consistency restoration* to restore consistency when the source or target changes; and *consistency checking* to check that a source and target are consistent. It is sometimes desirable to perform these operations incrementally, to save time or to minimize the side-effects of changes.

We refer the interested reader to Anjorin et al. [4] for an overview of industrial projects where different combinations of these consistency management operations were applied.

3 Alignment and Witness Structures

TGG tools also construct and use auxiliary structures to derive consistency management operations. In this paper, we concentrate on two primary structures: (i) an explicit representation of the correspondences between source and target elements, and (ii) a structure describing derivations by which TGG rules can be applied to generate a given triple.

We use the term *Horizontal Alignment* for (i). While (HA) can sometimes be computed based on unique identifiers or some other convention (elements with the same name correspond to each other), in general and for our running example, one must record correspondences. Figure 4 depicts a consistent pair of source and target models, together with a (HA) indicated as dashed double-headed arrows connecting family members to persons. For consistency management operations such as model synchronisation, the desired manner in which consistency is to be restored depends on the (HA). Consider, for example, handling multiple Bart family members in different Simpson families. If one of the Barts is now deleted or renamed, this change must be propagated to the *correct* person. TGG tools guarantee this using (HA).

TGG tools differ with respect to how "rich" (HA) is. A *minimal* set of correspondences can be used, e.g., connecting only family members to persons, or a *maximal* set of correspondences including connections, e.g., between families and persons connected to the family members of the family. Maximal (HA) can be automatically derived from TGG rules by simply creating all possible connections between all

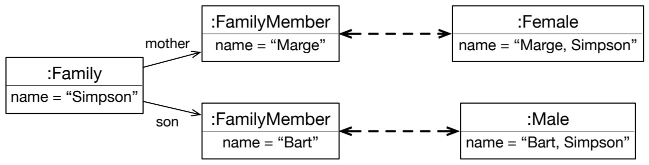

Figure 4. Horizontal Alignment for a Consistent Pair

elements in each rule every time it is applied. (HA) can be *strongly typed*, e.g., using different a specific correspondence type to connect mothers with females, and another type to connect daughters with females, or *weakly typed*, e.g., using the same type for all correspondences. (HA) can support correspondence nodes/links with *attributes*, allow *connections* (edges) between correspondence nodes, and enforce a 1-1 or 1-many connection between correspondence types and TGG rules (can a specific correspondence type be created by more than one TGG rule?). Finally, although all diagrams suggest that correspondence links are 1-1 connections, this is not necessarily true for all TGG tools. Some TGG tools connect multiple nodes (and edges) in the source and target models with a single correspondence "tentacle".

For the second auxiliary structure (ii), representing the sequence in which TGG rules can be applied to generate a given triple, we propose to use the term *Witness Structure (WS)* as used by Cheney et al. [5] or McKinna [14] as it *witnesses* the consistency of the triple. A triple is consistent if and only if such a witness exists. While some consistency management operations such as consistency checking produce a (WS) as their output, others such as model synchronisation can exploit a (WS) by equating consistency restoration with *repairing* such a (WS). If local repair operations can be performed cheaply, this allows for efficient consistency restoration as opposed to attempting to essentially recreate the (WS) from scratch. Figure 5 depicts a (WS) for the consistent triple depicted in Figure 4. It tells us that the triple is indeed consistent because it can be created by applying `CreateMother` then `AddSon` as indicated. The application of `AddSon` requires an element created by `CreateMother` as context and thus "depends" on the previous rule application (indicated in the figure as a `dependsOn` arrow.

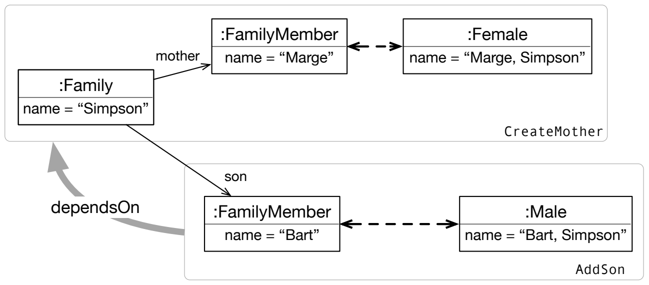

Figure 5. Witness Structure for a Consistent Pair

Note that the (WS) is typically not unique: in this example a different (WS) could be to create Bart first with `CreateSon` and then add Marge with `AddMother`. As with (HA), TGG tools differ with respect to how "rich" their (WS) are: Are dependencies recorded only on the level of rules (as suggested in Fig. 5)? Or also between elements, e.g., keeping track of the fact that the family member Bart depends on the family member Marge? In cases where a choice was made between conflicting rules – are all alternatives also maintained in the (WS)?

Finally, some TGG tools combine (HA) and (WS) into a single data structure for efficient storage and bookkeeping. Other tools separate the structures as some operations such as model transformation only require (HA) but not (WS).

4 Provenance and Explanations for BX

As the (technical) design space for (HA) and (WS) is large – as can be seen from the wide range of choices made in different TGG tools – it is helpful to concentrate on *what* these structures represent and not *how* they accomplish this. To provide a high-level characterisation of (HA) and (WS) as employed by TGG tools, therefore, we propose to view these structures as different types of *provenance* [5]:

(HA) is why-provenance: The existence of a (HA) that completes a pair of source and target models to a triple that the underlying TGG can generate answers the question *why* the pair is consistent.

(WS) is how-provenance: How can we *prove* that a pair of source and target models is consistent? The (WS) shows which rules and in what order they can be applied to generate a consistent triple extending the pair by a (HA). Note that (WS) implies (HA), and that (WS) can always be (re-)computed, albeit at a high cost. Both these observations are consistent with prior work concerning the relationship between why-provenance and how-provenance in databases [5, 9].

Apart from achieving conceptual clarity by abstracting from the (technical) details and differences of how (HA) and (WS) are implemented in different TGG tools, how else is our proposed characterisation of (HA) and (WS) as provenance useful? One observation it leads to is that (HA) and (WS) are used today either for correctness (if (HA) cannot be uniquely recomputed), or for efficiency (avoid revisting the entire triple by repairing (WS) locally as far as possible). For these two use cases, correctness and efficiency, it makes sense to keep provenance as minimal as possible – basically reduced to just what is absolutely necessary.

As TGG tools attempt to address richer and more challenging consistency management scenarios such as model *integration* (both models can be manipulated concurrently) [3], interacting and consulting with users will become increasingly important to enable conflict resolution and cope with the inherent non-deterministic nature of the consistency restoration process in this case. Such a communication with users, however, will require adequate *explanations* of the choices (to be) made [16].

5 Open Questions and Research Agenda

We argue that providing and maintaining rich(er) provenance will play an important role in *explaining* consistency management operations. Some initial ideas we have identified of how this can be achieved are outlined in the following as pointers for future work:

What about where-provenance? While we claim that (HA) and (WS) have arisen naturally as why- and how-provenance, respectively, it remains unclear, however, if and what *where-provenance* relates to in the context of TGG-based consistency management. While why-provenance is about *relationships* and inferences, where-provenance is about *locations* and data values [5]. What is a location in a TGG setting? Can we explain "where" a string in the target model "came from" in the source, or vice versa?

Enriching (WS): Possible extensions of (WS) include (i) adding connections to the propagated changes (typically referred to as *deltas*), i.e., as an additional *reason* that led to the current state of the (WS), and (ii) adding connections to (user) decisions that led to the current (WS). For both these extensions, it might become necessary to keep track of *deleted* elements in addition to created and modified elements (TGG tools currently have no reason to do this).

Complete track of changes: Instead of just repairing or *rewriting* the (WS) when restoring consistency, it can be useful to record all "versions" of the (WS). Such information can be exploited and used to enable interactive "debugging" of consistency restoration.

Conversely, we also believe that provenance research may benefit from ideas or techniques in the bx or TGG communities. For example, the TGG notion of "triple graph", with source data, target data, and (HA) structure linking them, seems to generalize several familiar models of annotation or trace structures used in database and workflow provenance. This observation suggests a strategy for importing ideas or technology from the bx community to the provenance settings, which may in turn lead to further opportunities for generalization or unification of provenance models:

Querying provenance and data Conventionally, provenance querying focuses on the (HA) graph and the source and target data are separate. Can we view (e.g.) database or workflow provenance as (HA) relating the source and target data (also represented as graphs) and derive query languages that support querying data alongside its provenance?

Provenance integration Currently, provenance models are often defined in a particular computational context, e.g. databases, workflow engines, operating systems, or scripts. Integrating provenance (and data) from these different systems is a challenge. Can such integration be facilitated by MDE ideas, using models and metamodels as an interlingua?

One provenance (semi)ring to rule them all? Since the introduction of the "provenance semirings" framework [9], database researchers have systematically explored algebraic structures that capture provenance for increasingly richer query languages [10]. Does an analogous notion of "most general" (WS) exist for TGGs?

We hope that this paper will lead to deeper discussion or collaboration between these two communities.

Acknowledgments The second author was supported by ERC Consolidator Grant Skye (grant number 682315).

References

[1] A. Anjorin. An introduction to triple graph grammars as an implementation of the delta-lens framework. In J. Gibbons and P. Stevens, editors, *Bidirectional Transformations - International Summer School*, volume 9715 of *LNCS*, pages 29–72. Springer, 2016.
[2] A. Anjorin, Z. Diskin, F. Jouault, H. Ko, E. Leblebici, and B. Westfechtel. Benchmarx reloaded: A practical benchmark framework for bidirectional transformations. In *BX*, pages 15–30, 2017.
[3] A. Anjorin, E. Leblebici, and A. Schürr. 20 years of triple graph grammars: A roadmap for future research. *ECEASST*, 73, 2015.
[4] A. Anjorin, E. Yigitbas, E. Leblebici, A. Schürr, M. Lauder, and M. Witte. Description languages for consistency management scenarios based on examples from the industry automation domain. *Programming Journal*, 2(3):7, 2018.
[5] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.
[6] J. Cheney, A. Finkelstein, B. Ludäscher, and S. Vansummeren. Principles of provenance (dagstuhl seminar 12091). *Dagstuhl Reports*, 2(2):84–113, 2012.
[7] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *ICMT*, volume 5563 of *LNCS*, pages 260–283. Springer, 2009.
[8] D. Dang and M. Gogolla. On Integrating OCL and Triple Graph Grammars. In *MODELS*, volume 5421 of *LNCS*, pages 124–137. Springer, 2008.
[9] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.
[10] T. J. Green and V. Tannen. The semiring framework for database provenance. In *PODS*, pages 93–99, 2017.
[11] S. Hildebrandt, L. Lambers, H. Giese, J. Rieke, J. Greenyer, W. Schäfer, M. Lauder, A. Anjorin, and A. Schürr. A Survey of Triple Graph Grammar Tools. *ECEASST*, 57, 2013.
[12] L. Klassen and R. Wagner. EMorF - A Tool for Model Transformations. *ECEASST*, 54, 2012.
[13] E. Leblebici, A. Anjorin, A. Schürr, S. Hildebrandt, J. Rieke, and J. Greenyer. A Comparison of Incremental Triple Graph Grammar Tools. *ECEASST*, 67, 2014.
[14] J. McKinna. Complements witness consistency. In *BX*, pages 90–94, 2016.
[15] A. Schürr. Specification of graph translators with triple graph grammars. In *WG*, volume 903 of *LNCS*, pages 151–163. Springer, 1994.
[16] P. Stevens. Is bidirectionality important? In *ECMFA*, volume 10890 of *LNCS*, pages 1–11. Springer, 2018.

A Glossary

Bidirectional transformations, BX Given two models, representing different (but related) parts of a system, a bidirectional transformation specifies (a) what it means for the two models to be consistent, and (b) given two models that have become inconsistent (or given a proposed change to one model), how to restore consistency.

Consistency checking Given a source and target model (and optionally a horizontal alignment or witness structure), check that they are consistent.

Consistency management/maintenance Restoring or establishing consistency between source and target models. The task of a bidirectional transformation. Includes model transformation, consistency checking, and consistency restoration.

Consistency restoration Given a consistent source and target model, and a (proposed) updated version or modification one of them, consistency restoration finds a new version of the other one that is consistent. In a TGG setting, this may also involve finding a new horizontal alignment or witness structure linking the new source and target.

Consistent For a TGG, a triple graph is consistent if it is derivable by the rules of the grammar. A source and target model pair are derivable if there is a horizontal alignment linking them that forms a consistent triple graph.

Horizontal alignment, HA An intermediate graph linking parts of the source and target models. Sometimes called correspondences, correspondence links/models, traces, traceability links, protocols, precedence structures. As TGGs can be viewed as an implementation of the more general delta lens framework for bx [1], we distinguish between horizontal alignment and *Vertical alignment* that refers to the representation of changes between models in the same domain (instances of the same metamodel).

Model In MDE, a data structure (usually, but not always, a UML-style graph).

Metamodel In MDE, a type or schema describing a collection of models (usually, but not always, a UML-style graph).

Model Driven Engineering, MDE Software engineering based on representing parts of a software system using *models*, or high-level data structures defining different aspects of the system.

Model generation Generating consistent triples by applying the rules of a TGG in some order (e.g. systematically or randomly). Analogous to string generation from a context-free grammar.

Model transformation Given a source model (or target model), to attempt to find a consistent triple that includes the given model.

Triple graph A graph structure with three parts, often called the *source model*, *target model*, and *correspondence model* a.k.a *horizontal alignment*. The source and target models are disjoint graphs. The correspondence model is a graph linking components of the source and target graphs to each other or to intermediate elements of the correspondence graph.

Triple Graph Grammar, TGG A grammar defining languages of triples of graphs (analogous to context free grammars for string languages). A grammar specifies start rules that describe possible initial states of the graph and additional rules that explain how to add structure to a partially-derived graph. Rules are usually written as graph triples themselves, with precondition structure shown in black and new node or edge structure shown in green or decorated with "++" (or both). Rules may also have side-conditions explaining how data values in source or target nodes should be related.

Witness Structure, WS An intermediate data structure describing one (or more) possible derivation(s) of a consistent triple graph. Other terms from (MDE) literature include rule dependency structure, precedence graph, and synchronisation protocol.