Journal of Visual Languages and Computing 38 (2017) 47–83

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

A systematic approach to constructing incremental topology control algorithms using graph transformation

Roland Kluge[a,*], Michael Stein[b], Gergely Varró[a], Andy Schürr[a,**], Matthias Hollick[c], Max Mühlhäuser[b]

[a] Real-Time Systems Lab, Merckstr. 25, 64283 Darmstadt, Germany
[b] Telecooperation Group, Hochschulstr. 10, 64289 Darmstadt, Germany
[c] Secure Mobile Networking Lab, Mornewegstr. 32, 64293 Darmstadt, Germany

ARTICLE INFO

Keywords:
Model-driven software engineering
Graph transformation
Graph constraint
Topology control
Static analysis
Correct by construction

ABSTRACT

Communication networks form the backbone of our society. Topology control algorithms optimize the topology of such communication networks. Due to the importance of communication networks, a topology control algorithm should guarantee certain required consistency properties (e.g., connectivity of the topology), while achieving desired optimization properties (e.g., a bounded number of neighbors). Real-world topologies are dynamic (e.g., because nodes join, leave, or move within the network), which requires topology control algorithms to operate in an incremental way, i.e., based on the recently introduced modifications of a topology. Visual programming and specification languages are a proven means for specifying the structure as well as consistency and optimization properties of topologies. In this paper, we present a novel methodology, based on a visual graph transformation and graph constraint language, for developing incremental topology control algorithms that are guaranteed to fulfill a set of specified consistency and optimization constraints. More specifically, we model the possible modifications of a topology control algorithm and the environment using graph transformation rules, and we describe consistency and optimization properties using graph constraints. On this basis, we apply and extend a well-known constructive approach to derive refined graph transformation rules that preserve these graph constraints. We apply our methodology to re-engineer an established topology control algorithm, kTC, and evaluate it in a network simulation study to show the practical applicability of our approach.

1. Introduction

Topology control (TC) is an important research area in the wireless network communication domain. TC aims at adapting the topology of wireless networks to optimize, for instance, the total power consumption, while maintaining crucial constraints of the topology (e.g., connectivity) [1–5]. A TC algorithm typically works by (i) first selecting a subset of the links of the original topology so that all required constraints are fulfilled, and (ii) then adjusting the transmission power of each node to reach its farthest neighbor across one of these selected links. In realistic settings, context events such as movement of sensor nodes continuously modify the structure of a topology. Therefore, a TC algorithm should operate in an incremental way by efficiently updating only the affected subset of links based on the occurred context events.

The development of a TC algorithm is performed by highly skilled and experienced professionals. The development process usually starts with an informal specification of the basic properties of a TC algorithm. This informal description is then supplemented by a formal specification using a theoretically well-founded framework such as graph theory [6–8] or game theory [9,10], which allows to prove that the algorithm preserves all required constraints. The first evaluation of a TC algorithm is typically carried out using a network simulator, which may be succeeded by a second evaluation in a testbed environment, i.e., on real wireless devices. Both types of evaluation require an implementation of the TC algorithm in one or—most often—two programming languages, such as Java or MATLAB for the simulation and C or C++ for the testbed evaluation [11]. This means that, in the end, the (hopefully) same TC algorithm is represented in two or three more or less completely different representations. In research publications, often only the formalization (along with the proofs of correctness and

[*] Corresponding author
[**] Principal corresponding author
E-mail addresses: roland.kluge@es.tu-darmstadt.de (R. Kluge), michael.stein@tk.informatik.tu-darmstadt.de (M. Stein), gergely.varro@es.tu-darmstadt.de (G. Varró), andy.schuerr@es.tu-darmstadt.de (A. Schürr), matthias.hollick@seemoo.tu-darmstadt.de (M. Hollick), max@informatik.tu-darmstadt.de (M. Mühlhäuser).

http://dx.doi.org/10.1016/j.jvlc.2016.10.003
Received 30 November 2015; Received in revised form 24 May 2016; Accepted 14 October 2016
Available online 29 October 2016

optimality) and a pseudo-code representation of the TC algorithm is given, while the implementations are typically omitted. Still, even for skilled researchers, it may be difficult to verify that the pseudo code is a valid implementation of the formal specification.

In the following, we illustrate this by means of the Cooperative Topology Control Algorithm (CTCA), proposed by Chu and Sethu in an IEEE INFOCOM paper in 2012 [12]. The authors first give a graph-based intuition of the proposed TC algorithm, which shall lead to an improved distribution of the nodes' lifetimes in a wireless sensor network [12], Section 3. The resulting goals are then formalized as so-called ordinary potential game [13], which allows to prove that the proposed algorithm eventually leads to a stable and optimal state of the network. The authors present an implementation of their algorithm in pseudo code, which is 83 lines long, distributed across four listings, and enriched with network-specific aspects such as communication message exchange; all these aspects make it highly non-trivial to understand the correspondences to the game-theoretic formalization [12], Section 5. In a simulation-based evaluation, the authors compare CTCA with other state-of-the-art TC algorithms. Unfortunately, no details about the simulation platform are given [12], Section 6. To the best of our knowledge, no testbed evaluation of CTCA has been performed yet.

While the previous example considers only one of the many existing TC algorithms, experience shows that it is (at least partly) representative. The example illustrates an obvious and prevalent gap between the formal specification, which serves for proving important properties of the algorithm, and the implementation, which serves for assessing the TC algorithm. Due to this gap, it inherently remains unclear whether the evaluated implementation of a TC algorithm fulfills the properties that have been proved based on the specification.

This is especially true for the case where an incrementally working TC algorithm is required. The transition from a batch TC algorithm, which takes a complete topology as input and produces a modified (optimized) complete topology as output, to an incremental version, which takes an arbitrary set of topology (context) modifications as input and produces a (minimal) set of topology adaptations as output, is an error-prone process. Experience shows that it is extremely challenging to cover all possible combinations of topology modifications in such a way that the computed topology adaptations never violate the given set of topology constraints and optimization goals. Contributions such as those by Zave show that even formalizations of well-known network algorithms often reveal special cases where these algorithms do not work properly [14,15]. For a more comprehensive survey of the application of formal methods to networking algorithms, we refer the reader to [16].

Towards a seamless construction process for TC algorithms. It is the vision of our research activities as part of the collaborative research center MAKI (Multi-Mechanism Adaptation for the Future Internet[1]) to close the gap between a carefully crafted formal specification and its derived implementation as follows. We propose a methodology for constructing TC algorithms starting with a concise formal specification and refining this specification step-by-step to an efficiently working implementation. The resulting implementation is correct by construction if we can show that all refinement steps preserve the properties of the initial specification. For this purpose, we adhere to a model-driven engineering (MDE) [17] approach, which works as follows:

- Topologies are formalized as models that are proper instances of a common meta-model that represents all relevant properties of the considered class of topologies, e.g., link-weighted topologies.
- The meta-model of a studied topology class is extended with a set of consistency constraints and optimization goals.
- Model transformation rules describe all relevant (context) modifications of a topology class and the expected constraint-preserving topology adaptations of the constructed TC algorithm.
- Code generators translate the rule-based description of a TC algorithm into an efficiently working implementation that can be used in a software simulator or a hardware testbed for evaluation purposes.

Directed or undirected graphs are commonly used to formalize the structure of communication system topologies (e.g., [6,7,12,18]) and TC algorithms are often sketched visually as sequences of topology graph modifications (e.g., [19,20]). Therefore, graph transformation (GT) constitutes a natural basis for developing our MDE methodology. GT offers a set of rule-based and declarative techniques for the high-level specification of model- or graph-manipulating algorithms with a well-defined semantics [21,22]. A variety of GT-based tools are available for formal specification and rapid prototyping purposes of specified algorithms [23–28].

GT languages and tools are established representatives of the whole class of visual languages (VL). As a consequence, our selected approach adheres to the tradition of both the VL and the MDE community to adopt visual modeling and programming languages for the high-level description of the structure and behavior of communication systems and, more generally, of distributed information systems. Today, UML [29]—an assembly of a number of previously popular VLs (e.g., state charts, message sequence charts, ROOM structure diagrams)—is a well-established visual modeling language used for MDE activities in the area of communication and distributed systems (e.g., [30–32]). Languages like eMoflon [23] or MechatronicUML [33] even integrate GT concepts for dynamic communication topology manipulation purposes with UML-like activity, class, and composite structure diagrams as well as state charts. Apart from these languages, the VL community has already been developing visual programming languages with well-defined syntax and semantics for distributed communication and information system construction purposes for several decades (e.g., G-Net [34,35]). A comprehensive survey of related research activities can be found in [36–38].

To summarize, the TC algorithm approach presented in this paper relies on the subclass of GT-based visual languages and follows the tradition of the formal program-construction-by-transformation approaches (see, e.g., [39]), which have their roots in research activities of the 1970s like the Munich project CIP (computer-aided intuition-guided programming) and are today part of the vision of model-driven engineering activities [40].

Model constraints may be used for specifying required or forbidden properties of models. Visual graph constraints have been introduced by the GT community as a means to characterize classes of graphs in a formal and declarative way [22,41,42]. For particular classes of graph constraints, formal refinement algorithms have been proposed that take a set of GT rules and graph constraints as input and produce a refined set of GT rules that *preserve* the given constraints [42,41,43]. More precisely, this means that applying the refined GT rules will not cause violations of the specified graph constraints.

A number of model-driven approaches for developing TC algorithms have been proposed (e.g., [44,45,32,46,47]) that target two major objectives: The first major objective is to reduce the complexity of developing TC algorithms from the point of view of domain experts by providing suitable (visual) abstractions, e.g., by using activity diagram-like syntax to specify the control flow of a TC algorithm. The second major objective is to simplify the testing and debugging by implementing the TC algorithm against a middleware layer, which enables that the exact same algorithm may be exercised inside a software simulation and a hardware testbed environment. However, to the best of our knowledge, none of these approaches focuses on integrating consistency properties *constructively* into the development process of TC algorithms. Instead, the proposed approaches focus on facilitating formal analysis, debugging, automated code generation, or

[1] In German: **M**ulti-Mechanismen-**A**daption für das **k**ünftige **I**nternet, http://www.maki.tu-darmstadt.de