Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Institutions for navigational logics for graphical structures[☆]

Fernando Orejas[a,*], Elvira Pino[a], Marisa Navarro[b], Leen Lambers[c]

[a] Universitat Politècnica de Catalunya, Barcelona, Spain
[b] Universidad del País Vasco (UPV/EHU), San Sebastián, Spain
[c] Hasso Plattner Institut, University of Potsdam, Germany

A R T I C L E I N F O

Article history:
Received 8 July 2017
Received in revised form 15 January 2018
Accepted 26 February 2018
Available online xxxx

Keywords:
Institutions
Graph logics
Navigational logics

A B S T R A C T

We show that a Navigational Logic, i.e., a logic to express properties about graphs and about paths in graphs is a semi-exact institution. In this way, we can use a number of operations to structure and modularize our specifications. Moreover, using the properties of our institution, we also show how to structure single formulas, which in our formalism could be quite complex.

1. Introduction

The extensive use of graphs in all areas of Computer Science is the reason for the relevance of being able to express graph properties and to reason about them. In particular, we are interested in the area of software modeling where, in the context of graphical modeling formalisms, like the UML, graph properties may be used to express constraints for a given model, and we are also interested in the area of graph databases, where graph properties may be used not only to express database constraints, but where a graph logic may be used as a basis to define a query language.

We may approach the problem of defining a graph logic in two different ways. On the one hand, we may just use a standard logic, after extending it with some graph concepts. For instance, Courcelle (e.g., [4]), studied a graph logic defined in terms of standard first-order (or monadic second-order) logic including some specific graph predicates. Similarly, in the area of graph databases, where foundational work has concentrated mainly on studying the expressivity or the complexity of classes of graph queries and other kind of related problems, they have studied extensions of first-order logic with classes of navigational path queries (see, e.g. [3,1]). On the other hand, we may define a specific logic where formulas include graphs (and graph morphisms) as first-class citizens, like in the *logic of nested graph conditions* (LNGC), introduced by Habel and Pennemann [6], which was proven to be equivalent to the first-order logic of graphs of Courcelle. A main advantage of LNGC is that it is generic, since it can be used for any category of graphical structures, provided that this category enjoys certain properties. If this is not the case we need a different encoding for each class of graphs. In addition, from a practical point of view, Pennemann [11] showed that a specialized prover for their logic outperformed some standard provers when applied to graph formulas using Courcelle's logic.

[☆] This work has been partially supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R, TIN2013-46181-C2-2-R) and from the Basque Project GIU15/30, and grant UFI11/45.

[*] Corresponding author.
E-mail address: orejas@cs.upc.edu (F. Orejas).

https://doi.org/10.1016/j.tcs.2018.02.031
0304-3975/© 2018 Published by Elsevier B.V.

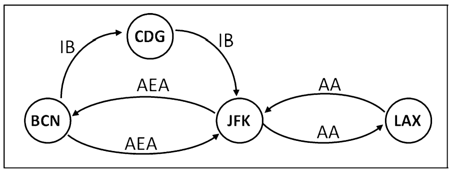

Fig. 1. A graph of connected airports.

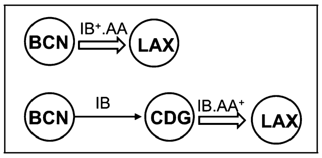

Fig. 2. Two connection patterns.

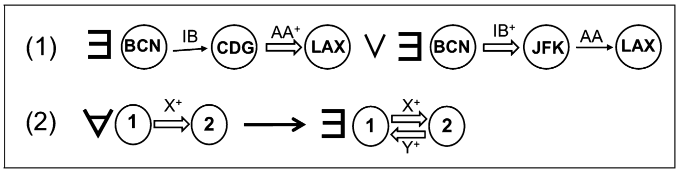

Fig. 3. Properties on airports networks.

A main problem of (first-order) graph logics is that it is not possible to express relevant properties like "there is a path from node n to n'", because they are not first-order. As a consequence, there have been a number of proposals that try to overcome this limitation by extending existing logics, like [7,12,8]. In particular, in [8] we extended the LNGC, allowing us to state properties about paths in graphs and to reason about them in a generic way (i.e. for arbitrary categories of graphical structures). Since this new logic allows one to describe properties of paths in graphical structures, we have called it a *navigational logic*.

Institutions were introduced in [5] to define the semantics of the Clear specification language, independently of any specification formalism. Showing that a given formalism is an institution allows us to use a number of constructions to structure and modularize our specifications [13]. For this reason, in this paper we show that a given navigational logic is a semi-exact institution. Moreover, using the properties of our institution, we also show how to structure single formulas, which in our formalism could be quite complex. For simplicity, in this paper we work with the specific category of labeled graphs, but the results can be generalized to arbitrary categories of graphical structures, following the lines of [8].

2. Navigational logics for graphical structures: introductory examples

The idea of our logics is that basic properties state if a given pattern is present or not in a graph. In our case, a pattern is like a graph but, in addition to normal edges, we may have other types of edges (depicted here as double arrows) representing paths between two given nodes. For example, let us suppose that we want to express some properties about graphs that represent networks of airports, like the one in Fig. 1, where each node is labeled with the name of an airport and each edge represents a direct flight between the source and target airports and is labeled with the name of the airline running the flight. Moreover, we assume that paths are labeled with regular expressions over the edge labels. In this context, in Fig. 2, we depict two patterns, where the first one represents a connection from BCN to LAX consisting of a sequence of IB flights followed by an AA flight, and the second one a direct flight from BCN to CDG followed by a connection from CDG to LAX consisting of an IB flight followed by a sequence of AA flights.

Formulas in our logics are built over patterns (and pattern morphisms) using quantifiers and the standard logical connectives. For example, in Fig. 3 we depict (in a pseudo-formal notation) two formulas to provide some intuition. The first one states that there must exist an IB flight from BCN to CDG followed by a sequence of AA flights leading to LAX, or there must exist a sequence of IB flights leading to JFK from BCN, followed by an AA flight to LAX. In the second one the long arrow denotes a morphism (the obvious inclusion morphism, in this case) between the pattern on the left (quantified universally) and the pattern on the right (quantified existentially), meaning that the target pattern is an extension of the source pattern. In particular, this formula states that for every connection between any two airports, there is a backward connection between them.[1]

[1] Here x and y represent any airline, i.e. $x, y = IB|AEA|AA| \ldots$.