Correctness of Graph Programs relative to HR$^+$ Conditions

Hendrik Radke
hendrik.radke@informatik.uni-oldenburg.de

Carl v. Ossietzky Universität Oldenburg, Germany[⋆]

Abstract. In (Pennemann 2009), the correctness of graph programs relative to nested graph conditions is considered. Since these conditions are expressively equivalent to first-order graph formulas, non-local graph properties in the sense of Gaifman are not expressible by nested graph conditions. We generalize the concept of nested graph conditions to so-called HR$^+$ conditions and investigate the correctness for graph programs relative to these generalized conditions.

Modeling of system states. As software systems increase in complexity, there is a growing need for design concepts that allow an intuitive overview of a system, usually by visual modeling techniques. Graph transformation systems are a visual modeling approach that emphasizes the interconnections of data structures. The states of a regarded real-world system are modeled by graphs, and changes to the system state are described by graph programs. Structural properties of the system are described by graph conditions.

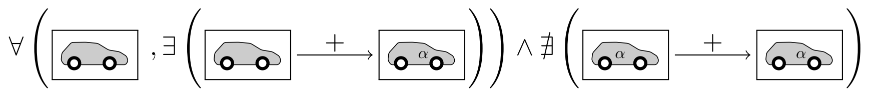
In [6], nested graph conditions are introduced. These conditions enhance first-order logic on graphs with a graphical representation of the nodes and edges involved. Nested conditions are expressively equivalent to first-order graph properties [10]. As such, they can express only local properties in the sense of Gaifman [4]. However, many real-world properties are non-local, i.e. they cannot be expressed by nested graph conditions. For instance, it is not possible to express the property "there is a path from node 1 to node 2", the connectedness or circle-freeness of a graph with these conditions, as these properties go beyond the k-neighbourhood for any node and any fixed k. Therefore, an extension is desired that can capture such properties.

HR$^+$ *conditions.* We propose to integrate hyperedge replacement systems with the nested graph conditions to form HR conditions [8]. The graphs in HR conditions are enriched with hyperedge variables, which are then replaced by graphs according to a hyperedge replacement system. Further enhancement to deal with subgraphs leads to HR$^+$ conditions. This way, non-local properties can be expressed by hyperedge replacement. In fact, HR$^+$ conditions are more expressive than monadic second-order formulas over graphs. The HR$^+$ condition $\exists(\overset{1}{\bullet}\xrightarrow{+}\overset{2}{\bullet})$

[⋆] This work is supported by the German Research Foundation (DFG), grants GRK 1076/1 (Graduate School on Trustworthy Software Systems).

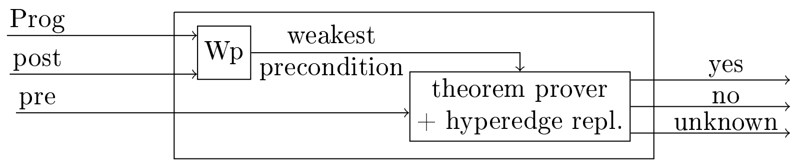
with the hyperedge replacement system $+ ::= \overset{1}{\bullet}\longrightarrow\overset{2}{\bullet} \mid \overset{1}{\bullet}\longrightarrow\bullet\xrightarrow{+}\overset{2}{\bullet}$ is satisfied for all graphs with a path between two nodes 1 and 2.

The following "car platooning" example after [9] may further illustrate the need for increased expressiveness. In order to save space and gas on highway lanes, cars travelling in the same direction drive in platoons, i.e. a tight row of cars with little distance in between. To ensure safety, the cars are partially controlled by a system adhering to a car platooning protocol. Each platoon has one and only one leader (designated by a small α) and an arbitrary number of followers. This property is representated by the following condition:



Model checking for HR$^+$ *conditions.* For finite graphs, the problem whether a graph G satisfies a HR$^+$ condition c is still decidable. In a naïve approach, one could, for every hyperedge in c, derive every graph permitted by the corresponding replacement system. The resulting conditions would be nested graph conditions, for which the satisfaction problem is decidable [10]. Since every hyperedge replacement system can be transformed into a montonous one [5] and no condition with a generated graph larger than G may be satisfied, the number of nested graph conditions to check is finite.

Correctness of graph programs relative to HR$^+$ *conditions.* HR$^+$ conditions can be used together with graph programs to build graph specifications in form of a Hoare triple {pre, Prog, post}. Our goal is to check the correctness of this triple, i.e. whether for all graphs G satisfying pre and all graphs H resulting from application of Prog on G, H satisfies post. Following an approach of Dijkstra [2], we construct a weakest precondition out of the program and the postcondition. The weakest precondition is constructed by first transforming the postcondition into a right HR$^+$ application condition for the program, then a transformation from the right to a left application condition and finally, from the left application condition to the weakest precondition. This is a generalization of the transformations from [7], where the hyperedge variables and the corresponding replacement systems have to be regarded. The correctness problem can thus be reduced to the problem whether the precondition implies the weakest precondition. It is planned to extend the ENFORCE framework [1] for nested conditions and its theorem prover component to support HR$^+$conditions.



Case studies. In order to show the practical merit of the results, several case studies will be performed.

- HR^+ conditions and graph programs shall be used to verify the car platooning protocol mentioned earlier [9].
- HR^+ conditions shall be applied to the problem of C++ template instantiation. Type checking of the templates is done on graphs, and type errors are output as graphs, including suggestions for remedies. This may help developers dealing with templates by giving clearer error messages for type errors in templates.
- HR^+ conditions shall be used to express and check OCL constraints for UML diagrams. Together with a transformation of UML metamodels into graph grammars [3], this allows the generation of instances of a metamodel with constraints.

References

1. Azab, K., Habel, A., Pennemann, K.H., Zuckschwerdt, C.: ENFORCe: A system for ensuring formal correctness of high-level programs. In: Proc. of the Third Int. Workshop on Graph Based Tools (GraBaTs'06). Electronic Communications of the EASST, vol. 1. 82-93 (2007)
2. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall, Englewood Cliffs, NJ (1976)
3. Ehrig, K., Küster, J.M., Taentzer, G.: Generating instance models from meta models. Software and System Modeling 8(4), 479–500 (2009)
4. Gaifman, H.: On local and non-local properties. In: Stern, J. (ed.) Proceedings of the Herbrand symposium: Logic Colloquium'81. pp. 105–135. North Holland Pub. Co. (1982)
5. Habel, A.: Hyperedge Replacement: Grammars and Languages, LNCS, vol. 643. Springer, Berlin (1992)
6. Habel, A., Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Structures in Computer Science pp. 1–52 (2009)
7. Habel, A., Pennemann, K.H., Rensink, A.: Weakest preconditions for high-level programs. In: Graph Transformations (ICGT 2006). Lecture Notes in Computer Science, vol. 4178, pp. 445–460. Springer (2006)
8. Habel, A., Radke, H.: Expressiveness of graph conditions with variables. In: Int. Colloquium on Graph and Model Transformation on the occasion of the 65th birthday of Hartmut Ehrig. vol. 30 (2010), to appear
9. Hsu, A., Eskafi, F., Sachs, S., Varaiya, P.: The design of platoon maneuver protocols for IVHS. Tech. rep., Institute of Transportation Studies, University of California at Berkeley (1991)
10. Pennemann, K.H.: Development of Correct Graph Transformation Systems. Ph.D. thesis, Universität Oldenburg (2009)