Checking Nested Graph Expressions with Alternating Finite Graph Automata

(Extended Abstract)

Berthold Hoffmann
Universität Bremen, D-28334 Bremen, Germany
hof@uni-bremen.de

Many application areas of computer science require to specify structural properties of graphs. *Nested (graph) conditions* [5], formulas that apply logical operators $\{\forall, \exists, \wedge, \vee, \neg\}$ to (injective) graph morphisms, are well-established to express such properties. However, nested conditions may only inspect subgraphs of bounded size, whereas one often wants to express "global" properties of a graph, like the existence of a path of unbounded length.

Orejas *et al.* [6] and Poskitt & Plump [7] have extended nested conditions so that global properties can be specified. These approaches have in common that these conditions are checked by logical means.

Here we propose another extension of nested conditions to specify global graph properties that can be checked by finite automata. The idea is as follows.

- An operation that concatenates graphs can be defined by gluing their designated front and rear nodes; this gives rise to *graph expressions* using concatenation, union, and Kleene star of graph languages, which are interpretations of regular string languages (over symbols with front and rear rank), defining a class of graph languages "that could be called regular" [4]. In particular, such a language may contain paths of unbounded length.
- Graph expressions can be transformed into equivalent *finite graph automata*, the transformations of which can then be made deterministic, by modest adaptations of the corresponding constructions for finite string automata [3]. The moves of a finite graph automaton consume an edge of the input graph. Different from string matching, several edges may match an input graph so that one of them has to be chosen nondeterministically, even if their transitions of the automaton are deterministic. Only if the automaton fulfils additional criteria, it may work without backtracking, in linear time.
- We introduce *nested (graph) expressions* as formulas that apply logical operators $\{\forall, \exists, \wedge, \vee, \neg\}$ to graph expressions (instead to graph morphisms). They allow global graph properties to be specified.
- Brugging, Hülsbusch, and König have lifted the notion of an *alternation* [2] from strings to graphs.[1] In [1], an *alternating graph automaton* has two kinds of states: if the automaton is in an existential state, at least one of its ongoing moves must lead to success; in a universal state, this must be the case for all of its ongoing moves. For our purposes, we adapt these automata so that they succeed if they recognize a subgraph of the input graph in question, not only if they recognize the entire subgraph.

[1] As an instantiation of finite automata on arrows in categories that have pushouts.

Submitted to:
GCM 2024

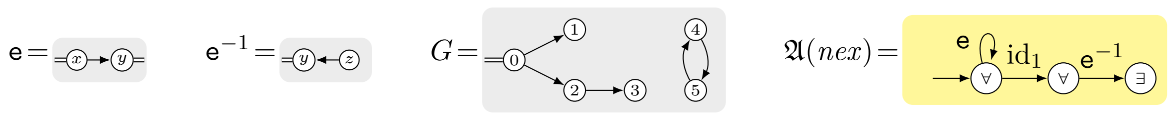

Figure 1: A nested graph expression and its alternating graph automaton

- Nested graph expressions can be checked with alternating finite graph automata that accept a given input graph if and only if it contains a subgraph that satisfies them. The translation takes the automata for the graph expressions contained in a nested graph expression (where final states are turned into universal states, and all others into existential states), and represents the operators of the nested graph expressions by corresponding kinds of states: Existential quantifiers and disjunctions are represented by existential states; universal quantifiers and conjunctions are represented by universal states; negated expressions $\neg nex$ are translated by taking the complement of the automaton for nex, where the complement of an alternating finite graph automaton is obtained by turning existential states into universal states and vice versa.

Figure 1 shows an example: The nested graph expression $nex = \forall(\mathtt{e}^*, \neg\exists\, \mathtt{e}^{-1})$ uses graphs $\mathtt{e}$ and $\mathtt{e}^{-1}$.[2] It specifies that in a graph like G, the unique front node (indicated by a double line to its left border) should be the root of a tree, i.e., connected to other nodes so that none of them is the target of other nodes. The alternating finite graph automaton $\mathfrak{A}(nex)$ checks this property. Like G, graphs satisfying nex may contain further nodes and edges unless they are connected to the subtree.

This research is in an early state. A detailed comparison to the approaches in [6, 7] is missing as well as an implementation of the alternating automata for nested graph expressions.[3]

Acknowledgments. The author thanks Annegret Habel for intensive discussions.

References

[1] H. J. S. Bruggink, M. Hülsbusch & B. König (2012): *Towards Alternating Automata for Graph Languages. Electron. Comm. of EASST* 47, doi:10.14279/tuj.eceasst.47.734.

[2] A. K. Chandra, D. Kozen & L. J. Stockmeyer (1981): *Alternation. J. ACM* 28(1), pp. 114–133, doi:10.1145/322234.322243.

[3] F. Drewes, B. Hoffmann & M. Minas (2024): *Finite Automata for Efficient Graph Recognition.* To appear in: *Post-Procedings of GCM 2023*, Available at arXiv:2404.15052.

[4] J. Engelfriet & J. J. Vereijken (1997): *Context-Free Graph Grammars and Concatenation of Graphs. Acta Informatica* 34(10), pp. 773–803, doi:10.1007/s002360050106.

[5] A. Habel & K.-H. Pennemann (2009): *Correctness of high-level transformation systems relative to nested conditions. Math. Struct. Comput. Sci.* 19(2), pp. 245–296, doi:10.1017/S0960129508007202.

[6] M. Navarro, F. Orejas, E. Pino & L. Lambers (2021): *A navigational logic for reasoning about graph properties. J. Log. Algebraic Methods Program.* 118, p. 100616, doi:10.1016/j.jlamp.2020.100616.

[7] C. M. Poskitt & D. Plump (2023): *Monadic second-order incorrectness logic for GP 2. J. Log. Algebraic Methods Program.* 130, p. 100825, doi:10.1016/j.jlamp.2022.100825.

[2] Nested condition $\forall(P,C)$ specify that that all graphs satisfying the premises P shall satisfy the conclusion C.
[3] However, M. de Rosa and M. Minas will present an implementation of plain graph expressions at this workshop.