Assessing computer science exercises using graphs

José Paulo Leal zp@dcc.fc.up.pt and Rúben Sousa up201001961@fc.up.pt

Graphs are mathematical structures that model relationships among objects. They can be used in a wide range of domains such as network topology, software architecture, digital circuit design, just to mention a few. Diagrams are an apt example of a document type with a graph-based representation that requires automatic assessment. However, graphs can be used for assessing exercises where the relationship among parts is important but not determinant. Finite Deterministic Automata (FDA) and even programming languages are examples of this kind of assessment.

The assessment of an FDA should be twofold, based on the recognized strings and on the structure of the state automaton. If an FDA recognizes all the strings it should, and only those, then it must be correct. Otherwise, examples of strings that should be recognized but are not, and vice versa, can be automatically generated. However, these examples seldom contribute to overcome the error. An helpful feedback must pinpoint what is wrong. For instance, what nodes are missing or what transitions must be removed. This can be achieved using graph assessment since the structure of an FDA can be represented by a state automaton.

Programming language assessment would also benefit from a similar approach. The standard way of assessing a program is to compile it and then execute it with a set of test cases. A program is considered correct if it compiles without errors and the output of each execution is what is expected. If it is incorrect then the most this approach can provide are examples of input that generates wrong output. It cannot pinpoint the errors in the code of the program. An attempt to make this kind of assessment should be based on the structure of the program, specifically on its abstract semantic graph.

The ultimate goal of this research is to define a general methodology for assessing graph-based exercises, applicable to a wide range of domains including FDAs, programming languages but also diagrams. The objective of this paper is to propose a graph assessment algorithm and to evaluate its efficiency for graphs with the size typically used in automated assessment. The distinctive feature of the proposed algorithm is that it uses the structure of the graph, rather than the its labels, to drive the matching process.

This methodology provides its own graph inspired data structure, adjusted to domains referred above. It is in fact a multigraph, meaning the a pair of nodes can be connected by several edges. Nodes and edges are typed and are characterized with sets of name-value pairs. Generic diagram editing software such as Visio or Dia persist diagrams as documents that can be easily parsed into this kind of data structure.

The main part of this methodology is a matching algorithm for the data structures that represent graph-based exercises. This algorithm is basically a graph homomorphism, i.e. a mapping on graph nodes that preserves its structure. Determining a graph homomorphism is an NP problem neither known to be solvable in polynomial time nor to be NP-complete. On the other hand, assessing a student graphs against a standard solution is not the same as determining a graph homomorphism, or if an homomorphism exists. In general, if the attempt is wrong there is no homomorphism. An assessment algorithm must determine the mapping that produces the highest grade and generate a correction.

Graph assessment may appear more complex than the graph homeomorphism problem. In a sense this true, since the graph homeomorphism problem is the special case where the all nodes and edges have have a single type and no properties. However, this is expected to be a rare case in exercises using graph-based languages. The approach is thus to use heuristics based on domain specific data - node and edge types and properties - to determine mappings for the largest possible graphs. It is important to characterize the graphs for which such approach is applicable, both regarding graph size (number of nodes), amount of difference between solution and attempt graphs.

Existing graph-based assessment systems use different approaches to deal with the high computational complexity of computing a mapping between node sets. Some researchers argue that graphs used in exercises are usually small and thus this complexity is not a serious problem. Since there are n! mappings for graphs with n nodes, finding the one that produces the best grade becomes a challenging task. An alternative to the a large number of mappings between node sets is to use labels. Most graph-based languages use nodes labels and these can be used determine a mapping. This approach either forces the student to use predefined labels or makes the assessment depend of computing the semantic similarity of labels.

The proposed assessment algorithm is based on the graph structure. This means that it actually computes the mapping between the node sets of both graphs that best preserves their edges. To avoid checking a large number of mappings it iterates over them testing the most promising first.The mappings are iterated in an order that allows the algorithm to prune the majority of them, when it is ensured that the remaining mappings cannot produce a better mapping. The iteration order is driven by the types and properties of nodes.