

Condensation Graphs - Software Testing

We are finally in a position to formalize an important simplification mechanism for testers.

Definition:

Give a graph $G = (V, E)$ its condensation graph is formed by replacing each component by a condensing node.

Developing the condensation graph of a given graph is an unambiguous (i.e., algorithmic) process. We use the adjacency matrix to identify path connectivity, and then use the equivalence relation to identify components. The absolute nature of this process is important: the condensation graph of a given graph is unique. This implies that the resulting simplification represents an important aspect of the original graph.

The components in our continuing example are $S_1 = \{n_1, n_2, n_3, n_4, n_5, n_6\}$ and $S_2 = \{n_7\}$. No edges can be present in a condensation graph of an ordinary (undirected) graph. Two reasons are:

- ? Edges have individual nodes as endpoints not sets of nodes. (Here, we can finally use the distinction between n_1 and $\{n_7\}$.)
- ? Even if we fudge the definition of edge to ignore this distinction, a possible edge would mean that nodes from two different components were connected, thus in a path, thus in the same (maximal!) component.

The implication for testing is that components are independent in an important way, thus they can be tested separately.