Correctness of Topological Sorting algorithm

Let’s start with defining $G'$ a subgraph of $G$ if it results by deleting some vertices and edges from $G$. A vertex $u$ precedes vertex $v$ if there is a directed path from $u$ to $v$. Node $u$ is a source if there is no vertex preceding it. In a DAG no vertex precedes another vertex that precedes it. Any subgraph of a DAG is a DAG and any non-empty DAG has a source vertex.

To prove that the execution of the body of the loop maintains an invariant we can use only the information about the state of computation “encoded” in the invariant. For the topological sort algorithm, we need to know that all the vertices in the list $L$ and the current graph $G'$ are exactly the vertices of the original graph $G$ (call this predicate $I_1$), that $L$ contains topologically sorted vertices from $G$ ($I_2$), and that no vertex in $G'$ precedes (in $G$) any vertex in $L$ ($I_3$). Finally, all vertices in $S$ are exactly the sources in $G'$ ($I_4$).

A useful invariant is the conjunction $I = I_1 \land I_2 \land I_3 \land I_1 \land I_4$. “Useful” because $I_1 \land I_4 \land \neg C$ imply empty $G'$ and thus all vertices of $G$ are in $L$, topologically sorted by $I_2$.

(i) The preamble establishes the invariant as $L$ is empty (trivially implying $I_1, I_2, I_3$) and $S$ the set of sources of $G$ ($I_4$).

(ii) The body of the algorithm:

1. Transfers a vertex $v$ from $S$ to $L$ deleting it from $G'$ and thus maintains $I_1$.

2. By $I_3$, $v$ does not precede any vertex in $L$, and thus appending $v$ to the end of $L$ maintains $I_2$.

3. Since $v$ was a source in $G'$, no vertex remaining in $G'$ precedes $v$ (and by $I_3$ any other vertex in $L$), thus $I_3$ is maintained.

4. Finally, any new source of $G'$ is added to all other sources in $S$, just maintaining $I_4$.

(iii) When $S$ is empty ($\neg C$), all vertices of $G$ are in $L$, which is topologically sorted by $I_2$. 