**Algorithm 1** Synthesize a DSPG from a non-DSPG

**Require:** Directed acyclic graph $G = (V, E)$
**Ensure:** DSPG-compliant graph $G' \subset G$

$pathsToProcess \leftarrow$ GetAllPaths($G$)
Sort $pathsToProcess$ by uncertainty or by length
$G' \leftarrow$ pathsToProcess.pop() ▷ pick the first and remove from the list
$nodesToPNPS \leftarrow \{\}$ ▷ Maps node to highest-nested PNPS it is part of
**while** $pathsToProcess \neq \emptyset$ **do**
    $currentPath \leftarrow pathsToProcess.pop()$
    $branch \leftarrow \emptyset$
    $endAddingPath \leftarrow false$
    **while** not $endAddingPath$ **do** ▷ Looking for the branch to try too add
        $edge \leftarrow currentPath.next()$
        **if** $currentPath$ is empty **then**
            $endAddingPath \leftarrow true$
        **end if**
        **if** $edge.sink$ is not in $G'$ **then**
            $branch.add(edge)$
        **end if**
        **if** $branch$ is not empty or $edge$ is not in $G'$ **then**
            $branch.add(edge)$
            $nodeSrc \leftarrow branch.source$
            $nodeTarget \leftarrow branch.target$
            $check \leftarrow dspgChecker(G', nodeSrc, nodeTarget)$
            **if** $check.bool = True$ **then**
                $G'.addEdges(branch)$
                **for** $node$ in $check.intermediateNodes$ **do**
                    $nodesToPNPS[node] \leftarrow smallerPNPS((nodeSoruce, nodeTarget), nodesToPNPS[node])$
                **end for**
                **for** $node$ in $branch.intermediateNodes$ **do**
                    $nodesToPNPS[node] \leftarrow moreNestedPNPS((nodeSoruce, nodeTarget), nodesToPNPS[node])$
                **end for**
            **end if**
        **else if** **then**
            $endAddingPath \leftarrow True$
            $branch \leftarrow [nodeSource, nodeTarget]$
        **end if**
    **end while**
**end while**