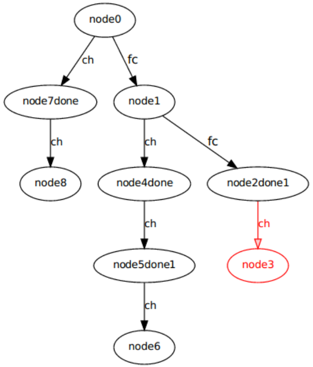
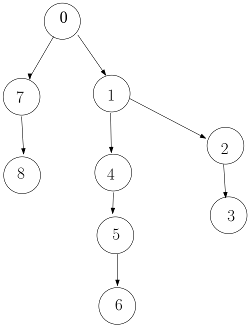
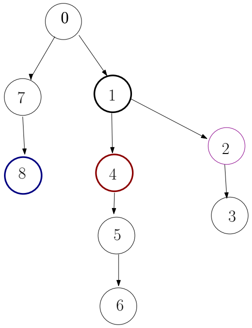


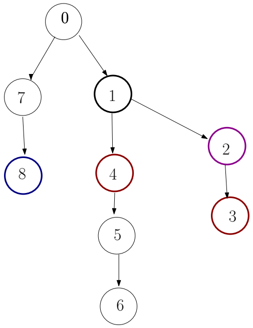
- for every function containing begin (see handle* functions in code)
- create a syncGraph with nodes defined by
- use of sync vars, begin fun, if else, loop etc.



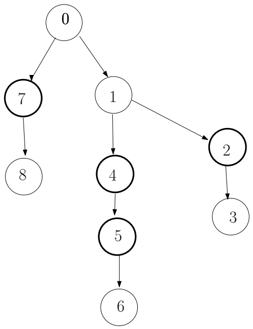
- For each node save the use of external variables (UseInfo). Here Node 1 & 2 uses external variable info.



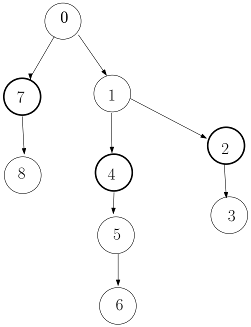
- For each use save the next Dominated (should reach there through all paths) sync node in same Function and Last Sync Node in the Function of definition.(see setNextSyncNode, setLastSyncNode)
- For Node 1 it is Node 4 and Node 7 respectively.



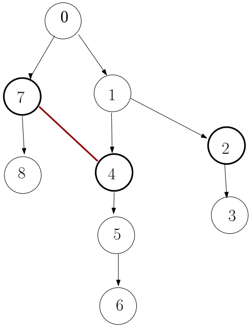
- For Node 2 it is Node 2 and 7 respectively
- Our aim is to check if Node 7 can be passed before Node 2/ Node 4 is passed.



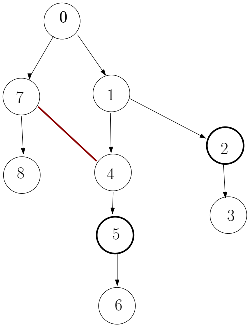
- The Graph has 4 sync points Nodes 2,4,5 7.
- Next id to find all possible set of Sync points that can be live at a given time. see(collectAllAvailableSyncPoints)



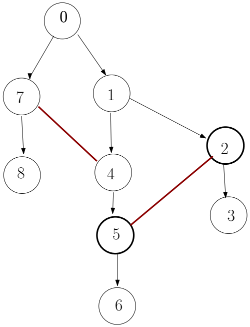
- Going through all paths we see 7,4,2 are the first set of sync points which are live.
- We add it to the list of possible sync points to set (see:VisitedMap).
- Now we try to see if there are sync Nodes that can be synchronized with each other. (see:threadedMahjong)



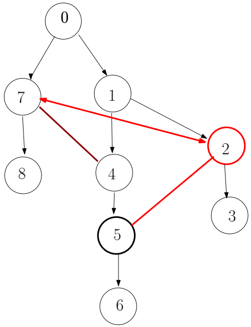
- Nodes 7 and 4 are matched.



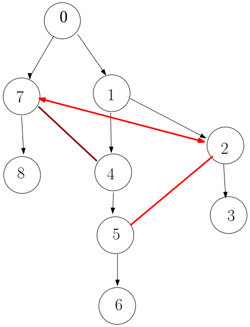
- Now Node 5 becomes available.



- Nodes 2 and 5 are Matched.



- if a newly matched set contains a node which is set as nextSyncPoint. we check if lastSyncPoint is already visited. If yes Throw error. Since node 7 is already visited we throw error at use of variable at node 2.



- No more sync point sets.
- delete and clean up.

- for If-else : both paths are taken each given a different VisitMap config. If any one of them leads to error. we show the warning.
- for Internal Functions: inline the nodes of internal functions. Break as soon as we see recursion.