```
// adapted from Braun, et. al.
writeVariable(variable, block, value):
   currentDef[variable][block] <- value

readVariable(variable, block):
   if currentDef[variable] contains block:
      return currentDef[variable][block]
   else:
      return readVariableFromPredecessors(variable, block)
   // ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   //   interesting bit here


readVariableFromPredecessors(variable, block):
   if block not in sealedBlocks:
      # this CFG is not complete, the block might gain a predecessor
      # thus the need for a phi is unclear, so let's assume it is
      # needed
      val <- new Phi(block)
      incompletePhis[block][variable] <- val
   else if block.preds.size() == 0:
      val <- undefined
   else if block.preds.size() == 1:
      # there is only one predecessor (and the block is sealed)
      val <- readVariable(variable, blocks.preds[0])
   else:
      # ok, let's search through predecessors and join them
      # with a phi instruction at the beginning of this block
      val <- Phi(block)
      writeVariable(variable, block, val)    # variable maps to new value
                                             # breaks cycles
      # vvvvvvvvv differs from paper, return value not used, so last
      # writeVariable is redundant along this path
      addPhiOperands(variable, val)

   writeVariable(variable, block, val)    # variable maps to value
   return val


addPhiOperands(variable, phi):
   for pred in phi.block.preds:
      phi.appendOperand(readVariable(variable, pred))


sealBlock(block):
   # get target variables of all incomplete phis in this block
```

```
   for variable in incompletePhis[block]:
      # for each variable, fill phi based on predecessors
      addPhiOperands(variable, incompletePhis[block][variable])
   sealedBlocks.add(block)


removeTrivialPhis(cfg):
   phis = gatherAllPhis(cfg)
   (trivial, workingSet) = splitTrivial(phis)

   while (trivial is not empty):
      for phi in trivial:
         removeTrivialPhi(phi)

      (trivial, workingSet) = splitTrivial(workingSet)
```