```
@With inter-procedural data-flow graph
analyzer InitializedVariablesAnalyzer
analysis direction:forward
lattice element type:Set<VariableWrapper>
use instructions fromDefOverride
construction parameters
  AnalysisResult<Map<VariableWrapper,Set<VariableWrapper>>> pointerAnalysisResult;

initial(program)->Set<VariableWrapper>{
    return InitializedVariablesAnalyzerHelper.getInitialVariableSet(program);
}

merge(program, input)->Set<VariableWrapper> {
    Set<VariableWrapper> result = InitializedVariablesAnalyzerHelper.getInitialVariableSet(program);
    modifying_iterator<Set<VariableWrapper>>iterator = input.iterator;
    while (iterator.hasNext) {
        result.retainAll(iterator.next);
    }

    return result;
}

fun(state, input, stateValues)->Set<VariableWrapper> {
    Instruction instruction = state.getInstruction();
    node<> source = (node<>)instruction.getSource();

    if (instruction.isStart()) {
        input.clear();
    }

    node<> sourceExpression = AnalyzerHelper.extractStatement(source);

    if (instruction instanceof WriteInstruction && ((WriteInstruction)instruction).getVariable() != null) {
        node<> var = (node<>)((WriteInstruction)instruction).getVariable();
        VariableWrapper targetVariable = newVariableWrapper(AnalyzerHelper.resolve(var));

        boolean omit=false;
        if (sourceExpression.isInstanceOf(IAssignmentLike) && AnalyzerHelper.isDereferenced(var,
                                         AnalyzerHelper.extractLeftSide(sourceExpression))) {
            omit = true;
        }

        if (!omit) {
            input.add(targetVariable);
        }

        if (sourceExpression.isInstanceOf(IAssignmentLike) && sourceExpression.@virtual == null) {
            node<Expression> left = sourceExpression:IAssignmentLike.getLValue();
            input.addAll(InitializedVariablesAnalyzerHelper.targets(targetVariable, instruction, pointerAnalysisResult,
                                         targetVariable.indirection-AnalyzerHelper.computeIndirection(left)));
        }
    } else if (instruction instanceof GeneratedInstruction) {

        GeneratedInstruction genInstruction=((GeneratedInstruction)instruction);
        VariableWrapper targetVariable = newVariableWrapper((node<>)(genInstruction.getParameter(0)));

        if (genInstruction.commandPresentation().startsWith("defInit")) {
            input.add(targetVariable);
        } else if (genInstruction.commandPresentation().startsWith("outInit")) {
            Boolean dereferenced = (Boolean)genInstruction.getParameter(1);
            if (targetVariable.indirection == 0 || dereferenced) {
                //this happens when a reference expression is used as an actual function argument
                input.add(targetVariable);
            }

            if (targetVariable.indirection > 0) {
                //this happens when a pointer typed variable reference is passed as an actual function argument

input.addAll(InitializedVariablesAnalyzerHelper.targets(targetVariable,AnalyzerHelper.getNonGeneratedPredecessor(g
enInstruction),
                                        pointerAnalysisResult, 1));
            }
        }
    } else if (instruction instanceof InterProcUnmapInstruction){
        node<>_variable = (node<>)((InterProcUnmapInstruction)instruction).getVariable();
        VariableWrapper variable = newVariableWrapper(_variable);
        input.remove(variable);
    }

    return input;
}
```