```
 * generated by Xtext 2.26.0
package dk.sdu.mmmi.mdsd.scoping

import dk.sdu.mmmi.mdsd.math.Assignment

/**
 * This class contains custom scoping description.
 *
 * See https://www.eclipse.org/Xtext/documentation/303_runtime_concepts.html#scoping
 * on how and when to use it.
 */
class MathScopeProvider extends AbstractMathScopeProvider {

    override IScope getScope(EObject context, EReference reference){
        var scope = super.getScope(context, reference)
        if(context instanceof VarUse){

            var IScope returnScope = null

            var letDefinition = EcoreUtil2.getContainerOfType(context, Local)
            var letVariable = EcoreUtil2.getContainerOfType(context, Assignment)
            if(letDefinition !== null && letVariable !== letDefinition.assignment){
                returnScope = addLetDefinition(letDefinition, context)
            }else{
                if(letDefinition !== null){
                    letDefinition = EcoreUtil2.getContainerOfType(letDefinition.eContainer, Local)
                }
                if(letDefinition !== null){
                    returnScope = addLetDefinition(letDefinition, context)
                }else{
                    returnScope = getVariableAssignmentsInScope(context);
                }
            }
            return returnScope
        }
        return scope;
    }
    protected def IScope addLetDefinition(Local letDefinition, EObject context){
        val containingLet = EcoreUtil2.getContainerOfType(letDefinition.eContainer, Local)

        if(containingLet === null){
            return Scopes.scopeFor(#[letDefinition.assignment],
getVariableAssignmentsInScope(context))
        }else{
            return Scopes.scopeFor(#[letDefinition.assignment], addLetDefinition(containingLet, context))
        }
    }

    protected def IScope getVariableAssignmentsInScope(EObject context){
        val root = EcoreUtil2.getRootContainer(context);
        val List<EObject> candidates = new ArrayList();
        //Get all variableAssignments
        for(VariableAssignment assignment: EcoreUtil2.getAllContentsOfType(root, VariableAssignment)){
            candidates.add(assignment as EObject)
        }

        //Should generate a list of all variables that are not let (ie the var difinitions that are global)
        val List<EObject> variableAssignments = candidates
        .filter(variable | variable !== EcoreUtil2.getContainerOfType(context, Variable))
        .toList()

        return Scopes.scopeFor(variableAssignments)
    }
}

```