Problem 9

1 The basic problem

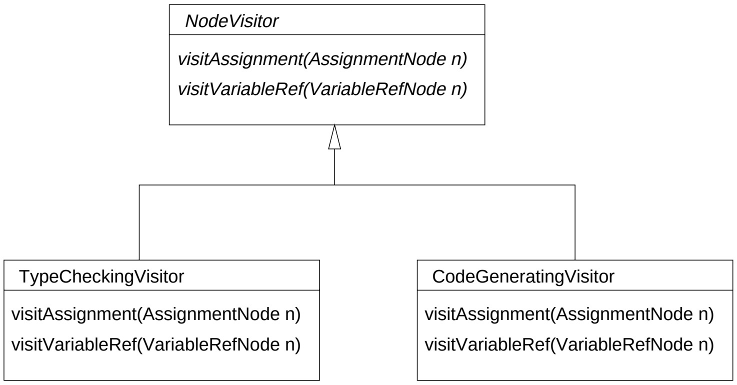
Suppose you have an integrated development environment (IDE) in which each kind of node in the abstract syntax tree is represented by a separate class; you have classes AssignmentNode, VariableRefNode and so on, all of which are specialised subclasses of an abstract class Node.

Now the IDE must provide many different operations on the program: for example, type checking, code generation, and pretty printing. Each operation needs, conceptually, to make requests of each node in the AST, with each kind of node responding differently to the request.

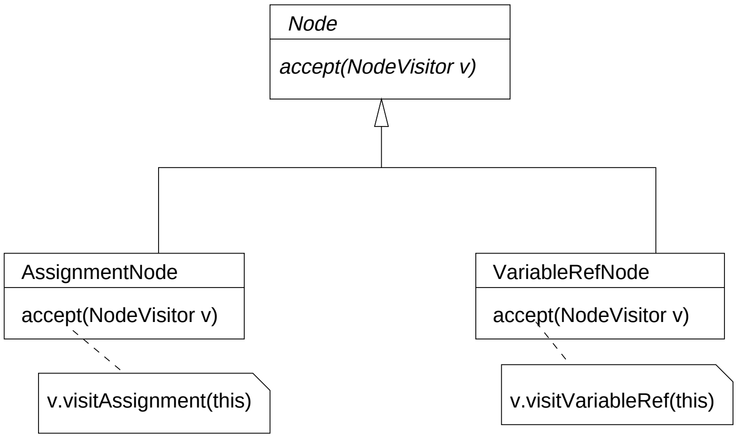
2 An initial solution...

3 A standard solution: VISITOR

We package related operations for each class into a separate object called a visitor: there is a concrete subclass of Visitor for each operation, e.g. TypeCheckingVisitor, CodeGeneratingVisitor:


NodeVisitor
visitAssignment(AssignmentNode n)
visitVariableRef(VariableRefNode n)
TypeCheckingVisitor
visitAssignment(AssignmentNode n)
visitVariableRef(VariableRefNode n)
CodeGeneratingVisitor
visitAssignment(AssignmentNode n)
visitVariableRef(VariableRefNode n)

To implement an operation – e.g. typechecking – the application creates a visitor of the right kind – e.g. a TypeCheckingVisitor. It then traverses the AST, sending to each node the message accept, with the visitor v as argument. Each node reacts by sending to v the visit message corresponding to its own type, *with itself as argument*:



The visitor implements, as its reaction to that message, the correct behaviour for performing this operation on this kind of node – e.g., typechecking an assignment.

4 When and why is this solution good?