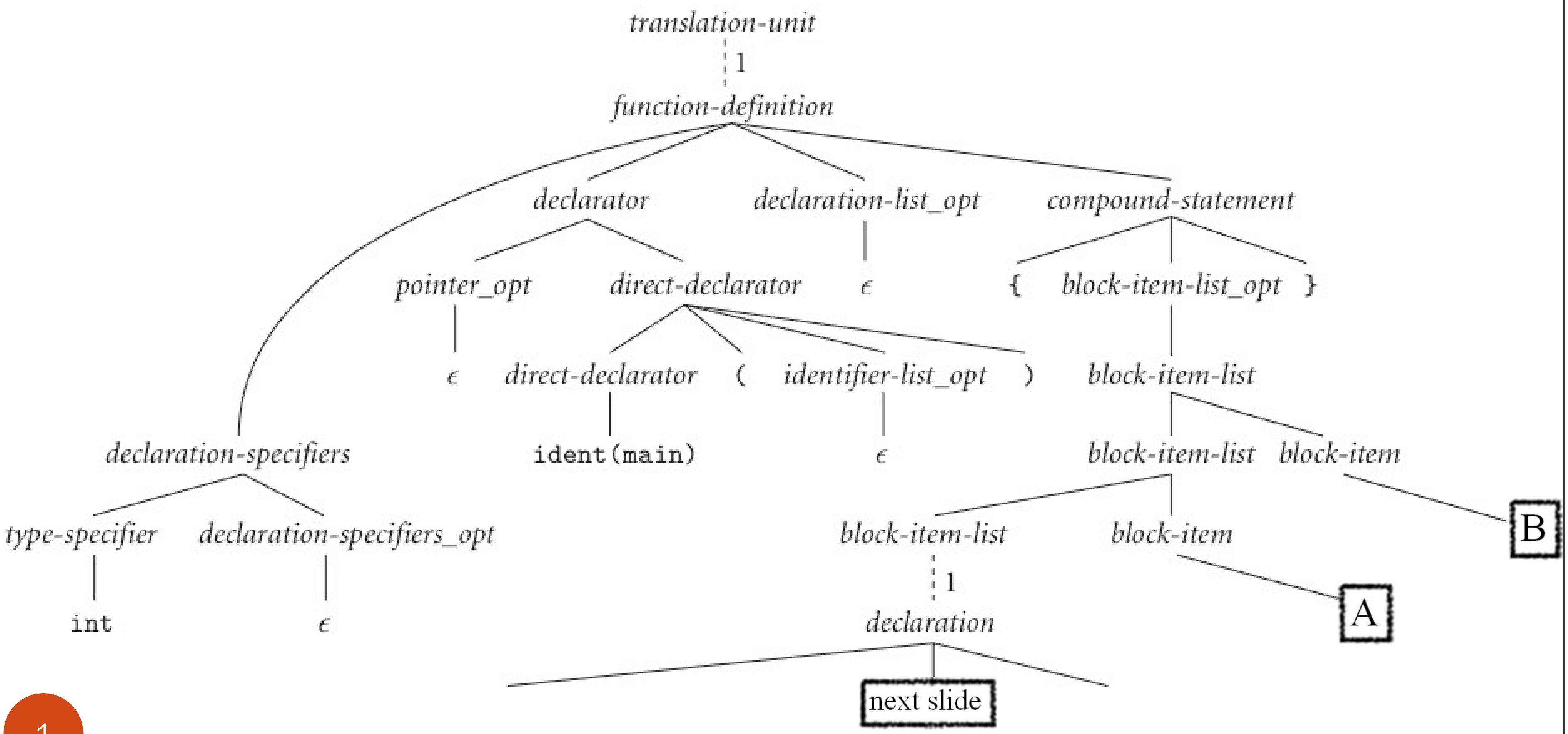
An Overview of Compilation

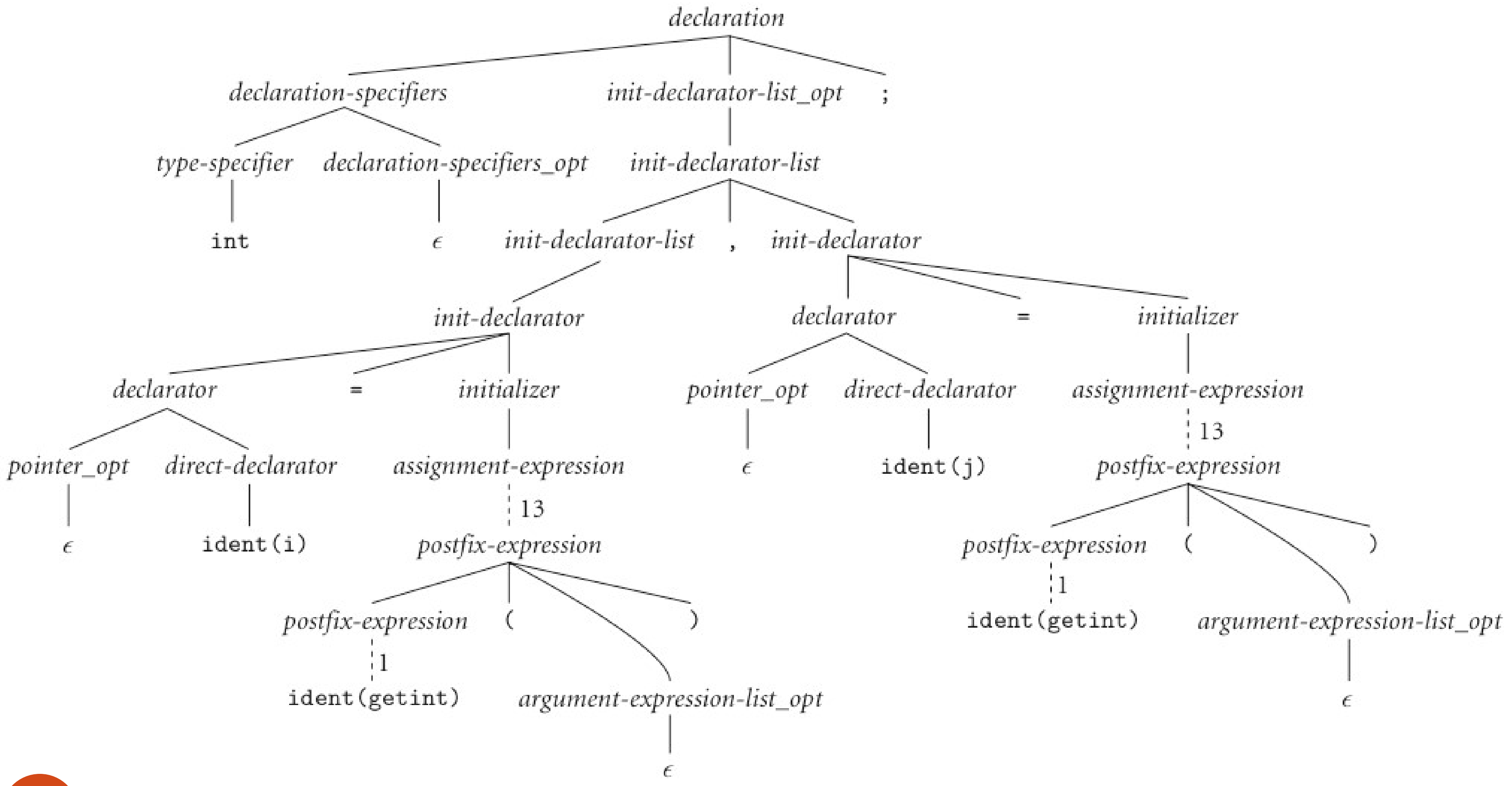
- Context-Free Grammar and Parsing
  - GCD Program **Parse Tree**:



(c) Paul Fodor (CS Stony Brook) and Elsevier

An Overview of Compilation

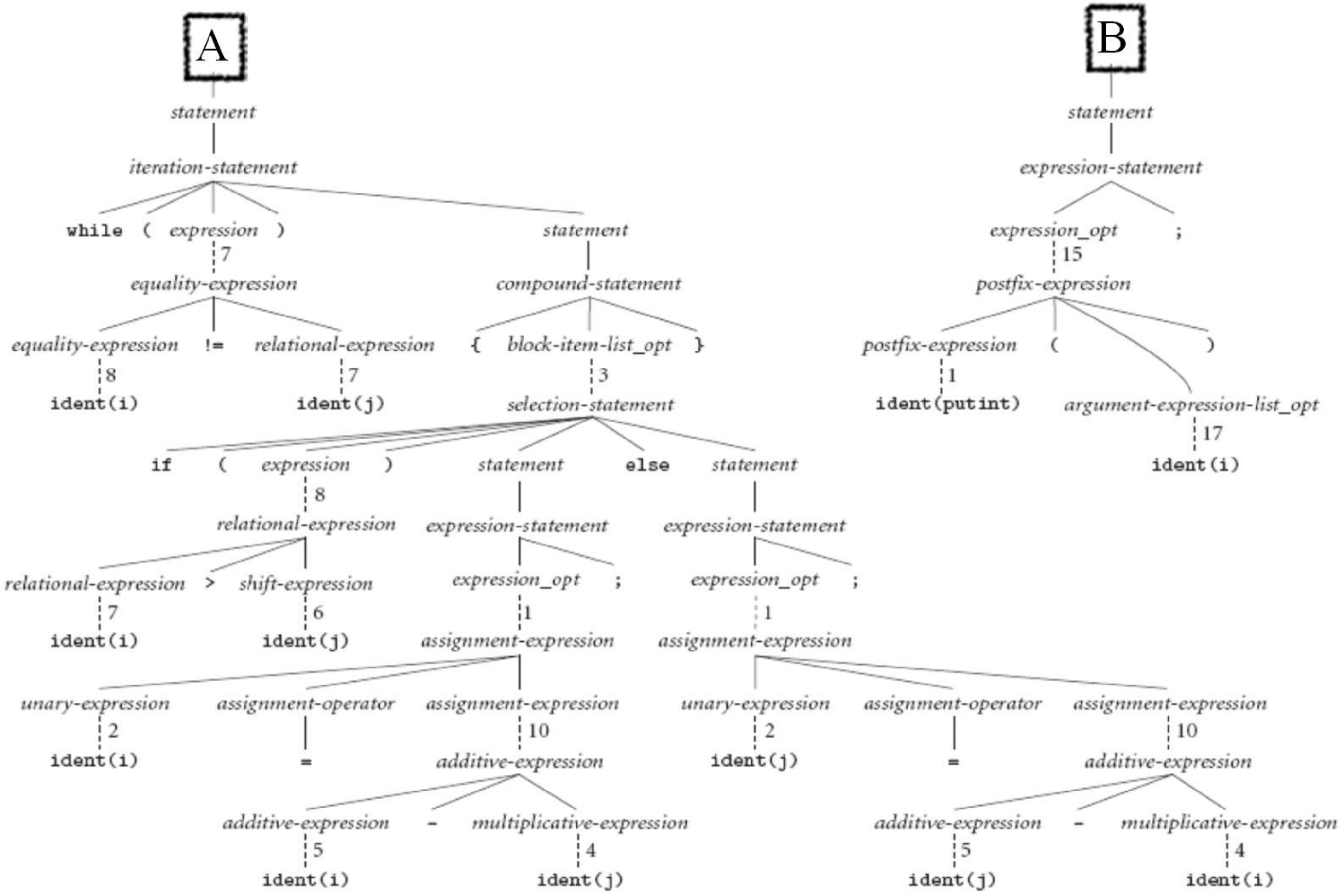
- Context-Free Grammar and Parsing (continued)



(c) Paul Fodor (CS Stony Brook) and Elsevier

An Overview of Compilation

- Context-Free Grammar and Parsing (continued)

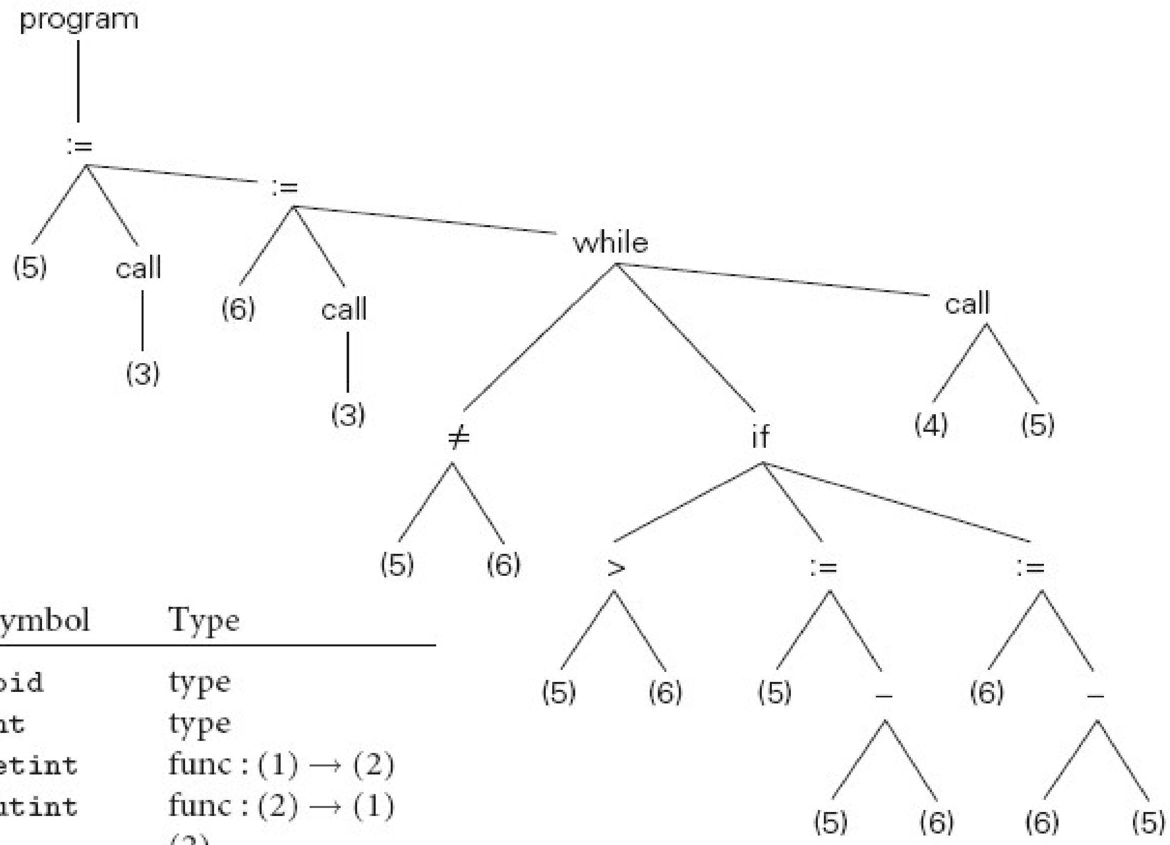


An Overview of Compilation

- Semantic Analysis and Intermediate Code Generation
  - **The parse tree is very verbose**: once we know that a token sequence is valid, **much of the information in the parse tree is irrelevant to further phases of compilation**
  - The semantic analyzer typically transforms the parse tree into an ***abstract syntax tree*** (***AST*** or simply a ***syntax tree***) by removing most of the "artificial" nodes in the tree's interior
  - The semantic analyzer also ***annotates*** the remaining nodes with useful information, such as pointers from identifiers to their symbol table entries
    - The annotations attached to a particular node are known as its *attributes*

An Overview of Compilation

- GCD Syntax Tree (AST)



Index	Symbol	Type
1	`void`	type
2	`int`	type
3	`getint`	func : (1) → (2)
4	`putint`	func : (2) → (1)
5	`i`	(2)
6	`j`	(2)