INTERMEDIATE REPRESENTATIONS

string table - external spellings

symbol/object table - attributes

parse tree - see Analysis Example

abstract syntax tree - see Synthesis Exampl

attributed tree - tree with nodes 'decorated' by attributes (or by pointers to symbol/object table)

DAG - directed acyclic graph; like tree except duplicate subtrees are shared

tuples - e.g. for A := B + C

[add, B, C, T_1]
[store, T_1, A, -]

(1) [add, B, C]
(2) [store, (1), A]

mixed forms

byte code

SYNTHESIS PHASES

Simplification and Translation to Virtual Machine

elaboration of representations and access paths
storage allocation
compile-time computation
elaboration of operators

Non-local Code Improvement (Optimization)

analysis and synthesis
equivalence-preserving program transformations
usually optional
rarely optimal

Code Emission

instruction selection
assembly code vs machine code
post-optimization
register allocation

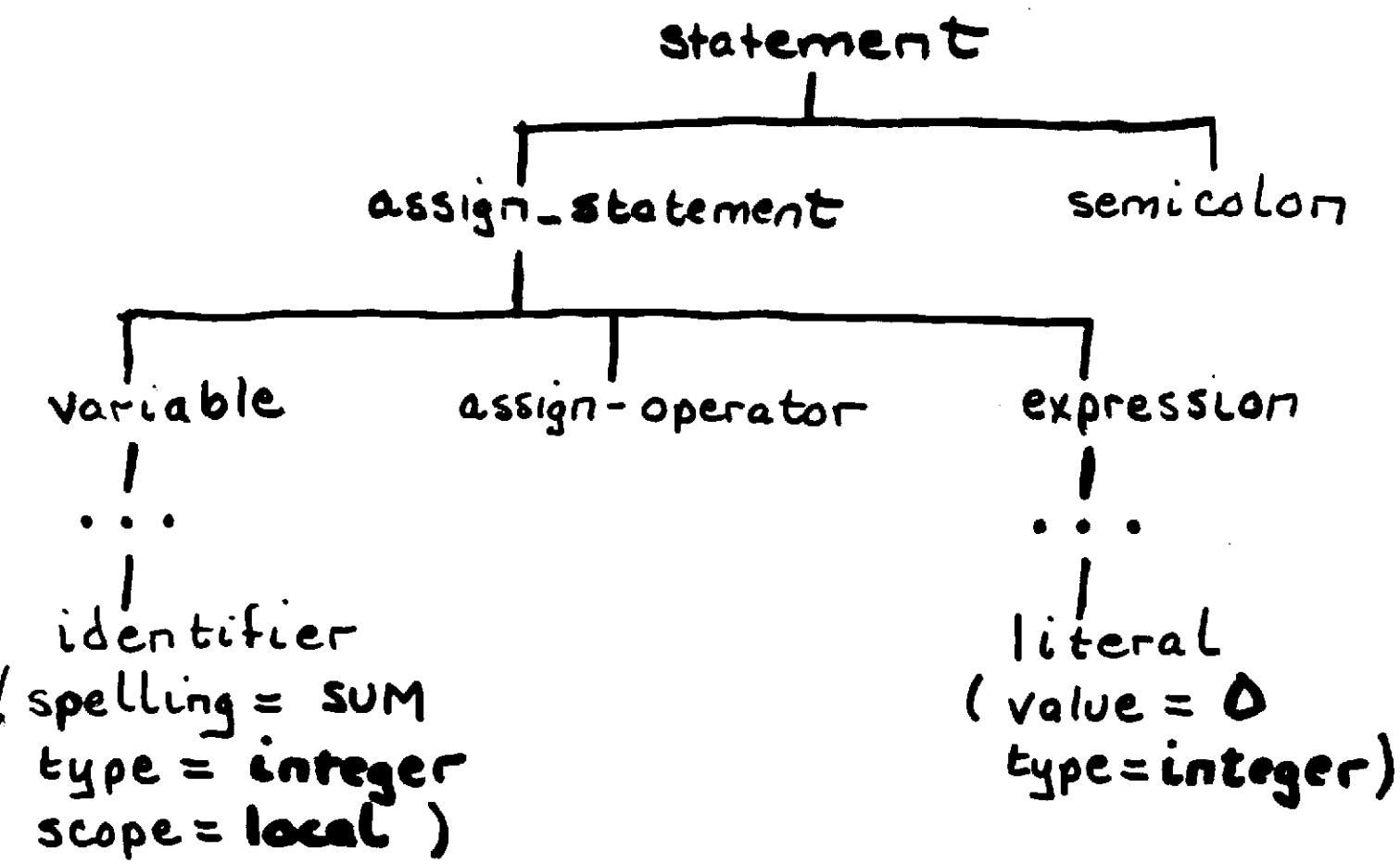
ANALYSIS EXAMPLE

SUM := 0 ; {INITIALIZE}

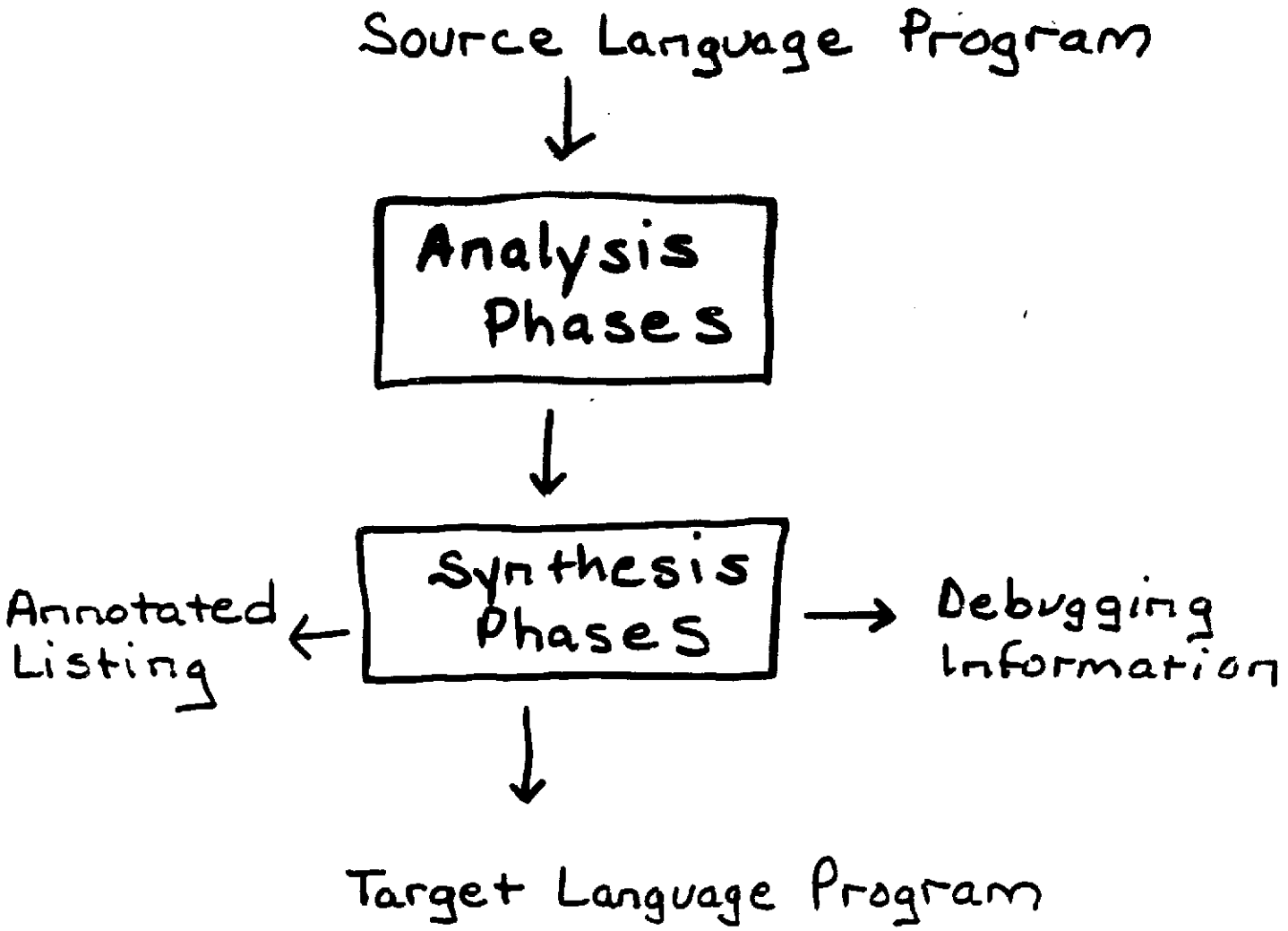
↓

identifier assign-operator literal semicolo

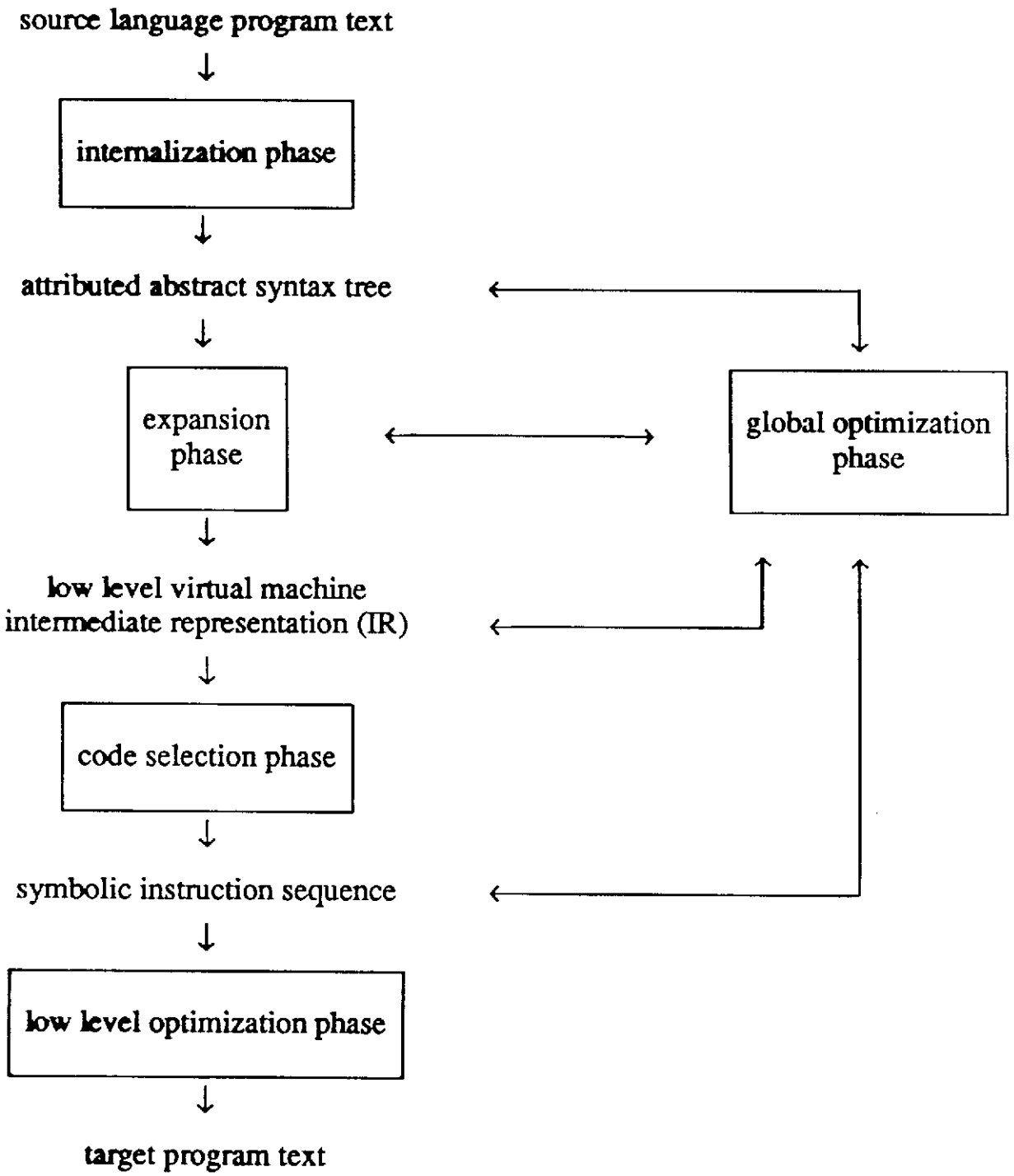
↓



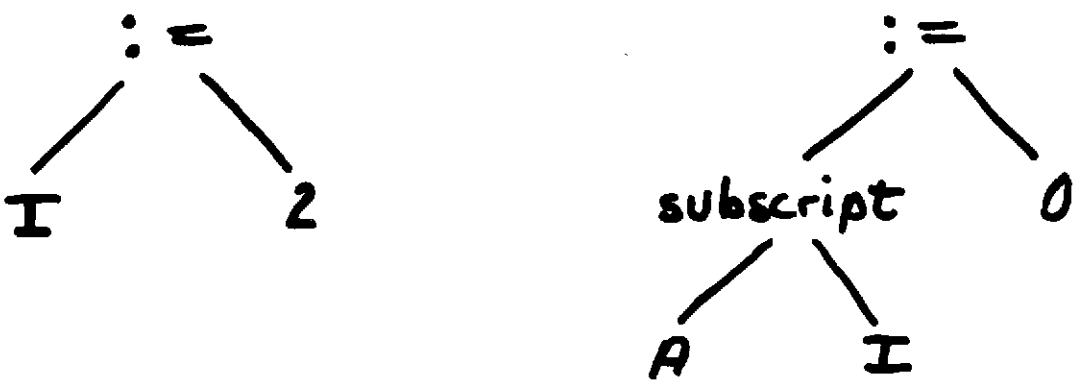
CONCEPTUAL COMPILER ORGANIZATION



The Synthesis Phases of a Translator



SYNTHESIS EXAMPLE



addr (I) ← val (2)

addr (A) + (val (I) - 1) * 4 ← val (0)

addr (A) + 4 ← val (0)

clear aloc + 4

Analysis Phases

Scanner (lexical analyzer)
maps text (sequence of characters)
into
sequence of tokens

Parser (syntax analyzer)
maps sequence of tokens
into
phrase structure

Semantic Analyzer (attribute evaluator)
associates attributes with tokens
and phrases
propagates information
disambiguates occurrences of token.
checks semantic rules of source
language

Aho, Sethi, Ullman

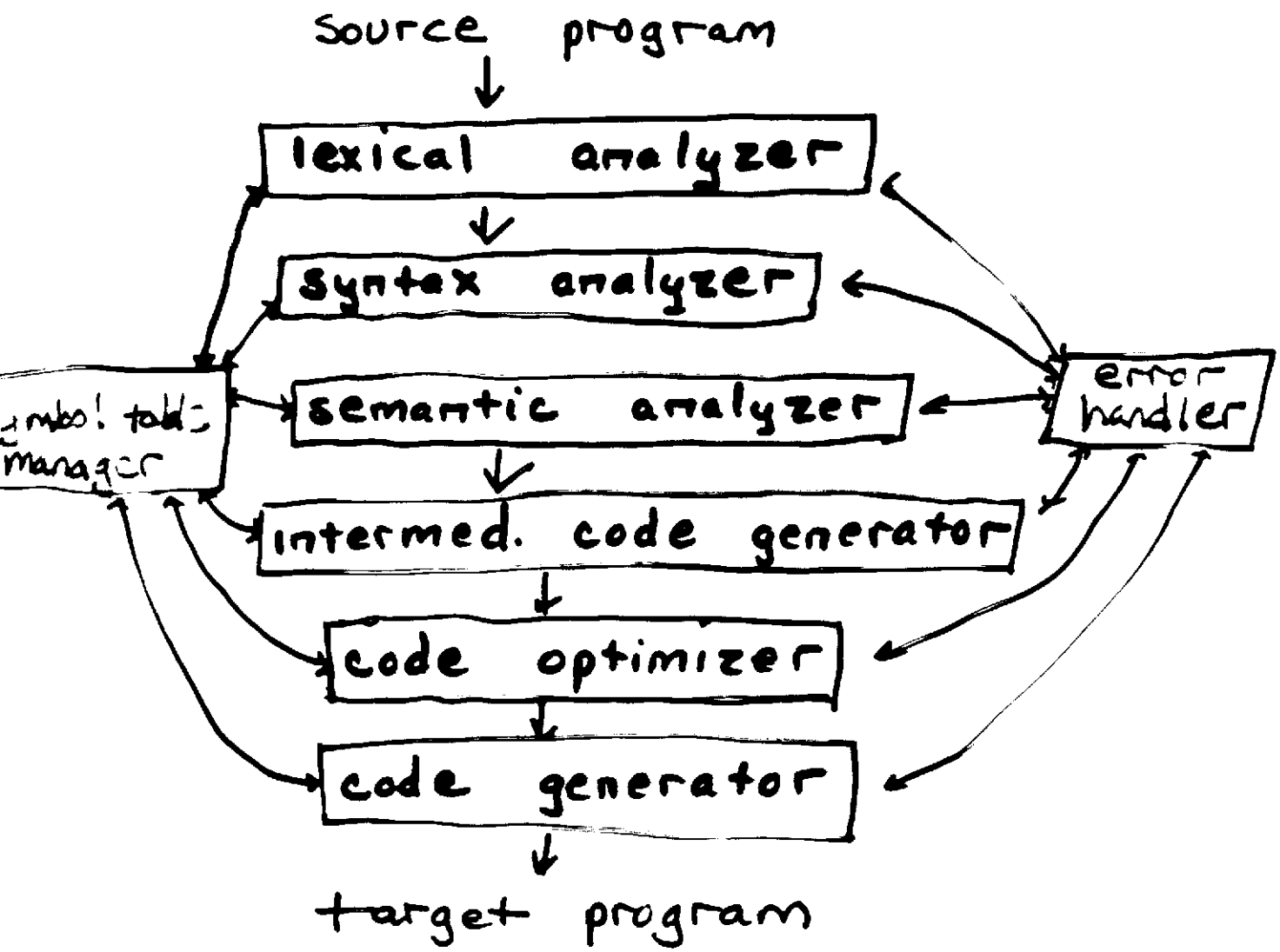


Fig 1.9 Phases of a Compiler