Q1 Define:
1. Synthesized Attribute
2. Inherited attribute
3. Dependency graph
4.Translation Scheme

Here are the definitions for each of the terms you requested:

1. Synthesized Attribute:
A synthesized attribute is an attribute of a non-terminal in a syntax tree that is computed from the attributes of its children in the tree. Essentially, it is an attribute whose value is determined by the attributes of its descendants, typically at the leaf nodes. Synthesized attributes are used to pass information upward in the syntax tree. For example, in a parse tree for an arithmetic expression, the value of an expression node can be the sum of the values of its operand nodes.
2. Inherited Attribute:
An inherited attribute is an attribute of a non-terminal that is computed from its parent or its siblings in the syntax tree. In contrast to synthesized attributes, inherited attributes flow downward in the syntax tree. They can also flow horizontally between sibling nodes. These attributes are used for propagating information from the top to the bottom or across different parts of the parse tree. For example, the type of a variable in a program may be inherited from its declaration.
3. Dependency Graph:
A dependency graph is a directed graph used to represent dependencies between attributes (synthesized and inherited) in an attribute grammar. In this graph, nodes represent attributes, and edges represent the dependencies between attributes. If attribute A depends on attribute B, then there is an edge from B to A. The graph is useful for determining the order in which attributes should be computed (topological sorting), ensuring that all dependencies are resolved in the correct order.
4. Translation Scheme:
A translation scheme is a mechanism used to define the semantics of a language's constructs in a compiler's syntax-directed translation. It involves associating actions (such as code generation or attribute evaluation) with grammar productions. In the translation scheme, each production in the grammar is augmented with embedded actions that perform operations when that production is applied. These actions guide the compiler on how to translate an input program into an intermediate representation, machine code, or some other target form.

Q2 Write syntax directed definition with inherited attributes for type declaration for list of identifiers. Show annotated parse tree for the sentence real id1, id2, id3.

A type declaration typically has the form type idlist, where type is a data type (e.g., real, integer) and idlist is a comma-separated list of identifiers (e.g., id1, id2, id3). We will first specify the grammar rules. Then, we'll define the semantic actions that propagate the type information (as inherited attributes) down to the list of identifiers.

Grammar Rules :
D → T L // Declaration: Type followed by list of identifiers
T → real // Type is 'real' (could extend to other types like 'integer')
L → id // Single identifier
L → L, id // Comma-separated list of identifiers

Syntax-Directed Definition
Let's denote:

- L.inhType: Inherited attribute for the list L, representing the type to be assigned to all identifiers in the list.
- id.type: Inherited attribute for each identifier, storing its type.
- T.type: the type, like real

The SDD with semantic rules is as follows:

Production	Semantic Rules
D → T L	L.inh = T.type
T → real	T.type = "real"
L → id	id.type = L.inh
L → id , L_1	L_1.inh = L.inh id.type = L.inh

2. Annotated Parse Tree for real id1, id2, id3
Now, let's construct the parse tree for the sentence real id1, id2, id3 and annotate it with the inherited attributes based on the SDD.

Input Sentence

```
real id1, id2, id3
```

Parse Tree Construction

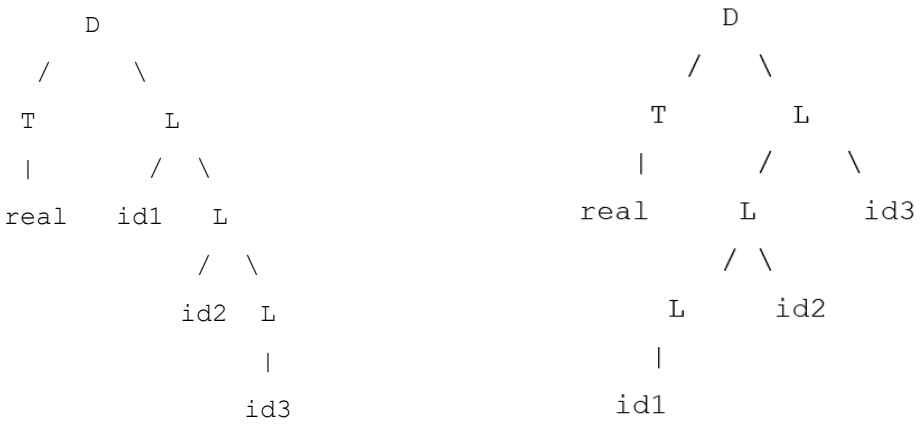
Using the grammar, we derive the sentence:

```
D → T L
T → real
L → id , L
L → id , L
L → id
```

Let's break down the derivation:

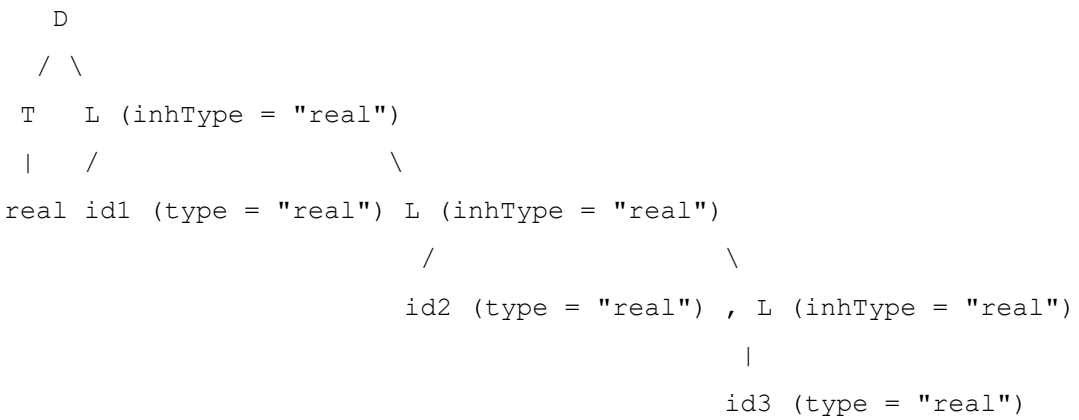
- Start with D → T L.
- T → real, so T produces the terminal real.
- For L, we apply L → id , L twice and then L → id:
    - First L → id , L for id1 , L', where id is id1.
    - Second L → id , L for id2 , L'', where id is id2.
    - Finally, L → id for id3.

The parse tree is:



Annotated Parse Tree

The parse tree with attributes is:



Attributes:

- T.type = "real"
- L.inh = "real" (passed down from T)
- id1.type = id2.type = id3.type = "real"