L-attribute definition, also known as an L-attribute grammar, is a formalism used to describe attributes associated with the production rules of a context-free grammar.

In L-attribute grammars, attributes are defined and computed in a bottom-up fashion, meaning that attribute values are synthesized or computed as the parser traverses the parse tree from the leaves to the root.

The idea behind this class is that, between the attributes associated with a production body, dependency-graph edges can go from left to right, but not from right to left hence L-attribute.

L-Attribute grammars are a special type of attribute grammars. They allow the attributes to be evaluated in one depth-first left-to-right traversal of the SDD. As a result, attribute evaluation in L-attributed grammars can be incorporated conveniently in top-down parsing.

Many programming languages are L-attributed. Special types of compilers, the narrow compilers, are based on some form of L-attributed grammar. These are a strict superset of S-attributed grammars. Used for code synthesis.

In L-attributed SDTs, a non-terminal can get values from its parent, child, and sibling nodes.

Some key characteristics and elements of L-attribute definitions:

1. Syntax: An L-attribute definition consists of a set of production rules, where each rule specifies a head (non-terminal symbol) and a body (sequence of symbols). The body of each rule may contain attribute references and semantic actions.
2. Attributes: Attributes represent properties associated with grammar symbols (terminals or non-terminals) or production rules. They can be used to store and compute information about the parsed input, such as values, types, or annotations.

3. Inheritance: L-attribute grammars allow the passing of attribute values from the children to the parent nodes in the parse tree. This is achieved through the use of attribute references in the production rules.
4. Semantic Actions: Semantic actions are pieces of code embedded within the production rules. They are responsible for calculating or assigning values to attributes during the parsing process. These actions may involve computations, assignments, or interactions with the attributes' values.
5. Bottom-up Evaluation: L-attribute definitions follow a bottom-up evaluation strategy. During the parsing process, attribute values are synthesized or computed at each node of the parse tree as the parser visits and processes the children of that node.
6. Dependency Graph: The dependencies between attributes form a dependency graph, which represents the order in which attributes must be evaluated to ensure correct attribute computation. Cycles in the dependency graph can lead to issues like circular dependencies or non-termination.

Example:

Let following production

```
S –>ABC
```

- S can take values from A, B, and C (synthesized).
- A can take values from S only.
- B can take values from S and A.
- C can get values from S, A, and B.
- No non-terminal can get values from the sibling to its right.

Attributes in L-attributed SDTs are evaluated by depth-first and left-to-right parsing manner.