saturday
6-06-2020

CD Assignment-3

UNIT-3

N.V.S.k. kalyani
1TJ19CA058
3rd B.Tech CSE-C

2 Marks

1 Mark

Synthesized Attribute for a parse tree node N is defined by a semantic rule in terms of attribute values at the children of N and N itself.

Inherited attribute for a parse tree node N is defined by a semantic rule in terms of attribute values at N's parent, N itself and N's siblings.

2 Mark

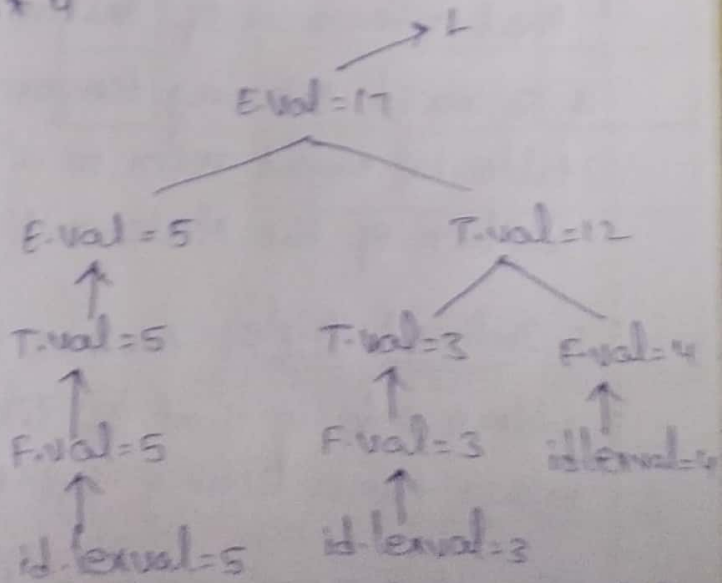
The Applications of SDT :-

- Tne Type checking
- Type conversion
- Identification of scope information
- Infix to postfix conversion
- Evaluation of expressions

3 Mark

Given string 5+3*4

$L \to E$
$E \to E+T \mid T$
$T \to T*F \mid F$
$F \to (E) \mid id$



4 Mark

A SDD is L-Attribute if each attribute must be either Synthesized and Inherited.

5a) A syntax directed translation scheme is a context free grammar with embedded semantic actions. In the SDD for our infix → postfix translation the parent either just passes on the attribute of its only child or concatenates them left to right and adds something at the end.

10 Marks

2a) 1. Quadruples:- A quadruple is a record structure with four fields, which are op, arg1, arg2 and result.

* The op field contains an internal code for the operator.
* The three-address statement x := y op z is represented by placing y in arg1, z in arg2 and x in result.

2. Triples:- Temporaries are not used and instead of that references of instruction results are used.

* To avoid entering temporary names into the symbol table we might refer to a temporary value by the position of the statement that computes it.

3. Indirect Triple:- Another implementation of three-address code in that of listing pointers to triples. rather than listing the triples themselves.

Ex:- d = a * minus b + b / minus c

Three-address code

t_1 = minus b
t_2 = a * t_1
t_3 = minus c
t_4 = b / t_3
t_5 = t_2 + t_4
a = t_5

Quadruples

op	arg1	arg2	result
minus	b		t_1
*	a	t_1	t_2
minus	c		t_3
/	b	t_3	t_4
+	t_2	t_4	t_5
=	t_5		a

Triples

	op	arg1	arg2
0	minus	b	
1	*	a	(0)
2	minus	c	
3	/	b	(2)
4	+	(1)	(3)
5	=	a	(4)

	op
35	(0)
36	(1)
37	(2)
38	(3)
39	(4)
40	(5)

Indirect Triples

	op	arg1	arg2
0	minus	b	
1	*	a	(0)
2	minus	c	
3	/	b	(2)
4	+	(1)	(3)
5	=	a	(4)

2b) Type checking :-

A compiler must check the type rules followed by the program being compiled.

Information about data types is maintained and verified by the compiler.

Type checkes is a module of a compiler devoted to type checking Tasks.

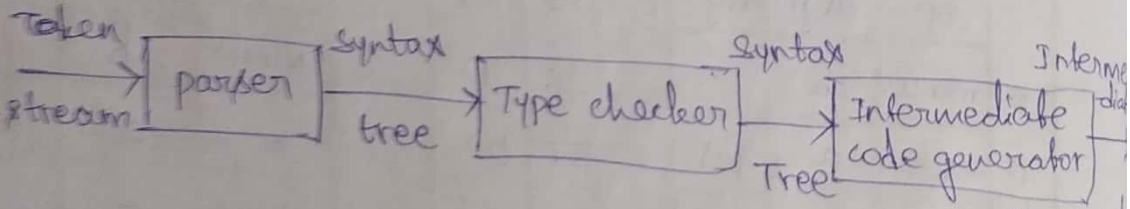
Type checking may be two types :-

1. Static Type checking : checking done at compile time
2. Dynamic type checking : checking done at run time

pascal and c language follows static type checking. It is used to check the program before its execution.

Some modern programming language like python follow Dynamic type checking.

The design of Type checker :- Type checker verifies each identifier and type expressions are following type rules or not.



position of Type checker

The design of Type checker depends on the following

* The Syntatic structure of language constructs.
* Type Expressions of the language.
* Semantic rules for designing types of constructs.

30) Backpatching :- In the process of code generation all the labels may not be known in a single pass hence we use a technique called Backpatching.

Backpatching is the activity of filling up unspecified information of labels using appropriate semantic actions during the process of code generation.

Boolean Expressions :-

1. They are used to compute logic values.
2. used as conditional Expressions in statements that flow of control.

* Boolean expressions are composed of Boolean operators such as and or & Not.
* Boolean operators are applied to Elements that are Boolean variables or relational Expression.
* Relational Expressions are of the form :

  E_1 relop E_2

  where E_1, E_2 are arthimetic Expressions.

  relop - relational operators $<, <=, >, >=, ==, !=$
* Boolean Expression are generated by the follow-ing grammer.

  $E \rightarrow E$ or $E \mid E$ and $E \mid$ not $E \mid (E) \mid$ id relop id/true/false
* order of preccedence : Not, and or.

3b) Methods of Translating Boolean Expression. There are two principle methods to represent the value of a Boolean Expression

1. Numerical Representation
2. Flow of control statements.

Numeric Representation :

* In this representation of Boolean Expression, It is used to denote true and 0 to denote false.
* Boolean Expression are evaluated from left to right.

Ex: t_1 : not b

t_2 : b and t_1

t_3 : = a or t_2

Translation scheme for producing three address code for Boolean Expressions:

In this scheme we are going to have the following assumptions.

* Exit
* nextstart

* Flow of control statements:

→ Consider the translation of Boolean expression into three-address code in the context of if-then, if-then-else and while-do statements.

$S \rightarrow$ if E then S_1,

| if E then S_1 else S_2

| while E do S_1

* In each of these production E is the Boolean Expression to be translated.
* Two labels are associated with Boolean Expression on E, They are:

→ E.True - The label to which control flows if E is true.

→ E.False - The label to which control flows if E is false.

a) Backpatching are three various types. Those are

1. makelist (i)
2. Merge (p_1, p_2)
3. Backpatch (p, i)

Ex:-

$E \rightarrow E_1$ or M E_2
| E_1 and M E_2
| not E_1
| (E_1)
| id_1 relop id_2
| True
| False

$M \rightarrow \varepsilon$

* Synthesized attributes truelist and falselist of non-terminal E are used to generate jumping code for Boolean Expression.
* As a code is generated for E jumps to true and false. Exists are left incomplete, with the label field unfilled.