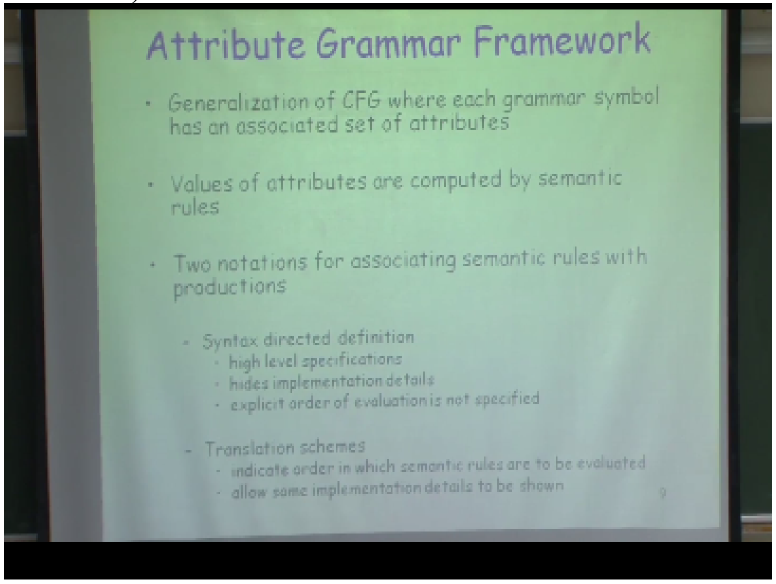
Indian Institute of Technology
Kanpur
NP – TEL

National Programme
On
Technology Enhanced Learning
<u>**Course Title**</u>
Compiler Design
Lecture – 14

by…
Prof. S. K. Aggarwal
Dept. of Computer Science and Engineering

Somebody please remain me at 4:45 stop and then discuss for the board of exam next week we have 1st paper exam there were some discussion in the beginning for open books open boards close boards whatever we discuss that finalize because then I need to spent with you so somebody please remain me at 4:45 so we will start with cemetery analyzer with compilation and look at the models how we are going to create a cemetery analyzer.

(Refer Slide Time: 00:51)



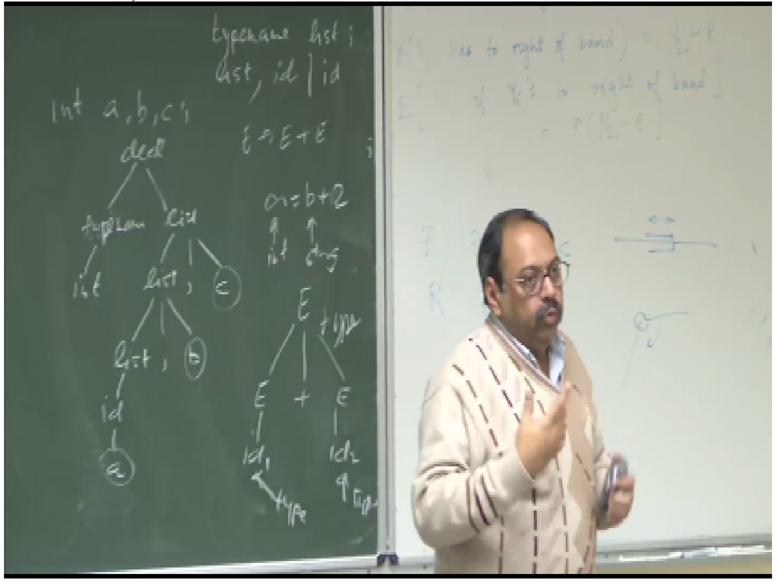
Attribute Grammar Framework

- Generalization of CFG where each grammar symbol has an associated set of attributes
- Values of attributes are computed by semantic rules
- Two notations for associating semantic rules with productions
  - Syntax directed definition
    - high level specifications
    - hides implementation details
    - explicit order of evaluation is not specified
  - Translation schemes
    - indicate order in which semantic rules are to be evaluated
    - allow some implementation details to be shown

9

And what was coming out perfect need to have certain property associated each of the grammar symbols and we are going to call that as attribute grammar so let us look at what is the framer for attribute grammar and how you are going to have an intimation okay so this is really a non nominal of the syntax analyzes everybody continue to do that basically what I want to do now is that once we have the concerti grammar and we have the grammar in this.

Each grammar symbol has an associated set of attribute this point of time let us not worry about what is competitor the bending on the what the time to do this property is could be different for example if I am type checking so what I am talking about is now more general thing if I am trying to type checking that I want that some point of time when I start doing fourth generation the later part of compilation then you will find that with certain attitudes with certain grammar symbols.
I would like to associate information about where these symbols are stored so I can say the Covanta associate a register with this expression among two stored this particular expression where registered or I want to store variable into certain register so ethiroot is various in generalized property with annual associate with each of the grammar symbols and remember that each grammar symbols can have more than one so it has several property not the small.
So this is really a generalization of conceptive grammar various symbols will have certain property associated and as far as tight checking is concern I would like to do certain computation so min the previous class you fall towards end I was giving an example.
(Refer Slide Time: 02:41)



That if I am trying to pass same the type I declaration and I want to bank and pass book declaration and how would I compute values and how would I compute and when I encounter C the five of the C is int okay and the kind of C being created for this pass that I will have a name here and I may have so my rule may look something like saying that a type name followed by a list okay.

This may be a rule and I then I will say the list is nothing but it could be list become a rule so the list followed by let us say ID separate by comma okay so I could get something like this typing I say this is type name this is list okay so this is what my declaration is and typing could be this is the integer this could be a list followed by comma followed by ID which I say C here and this again will be list followed by comma followed by ID B and list could be so let me have another rule let me say that list could be ID.

And list could be ID here and followed by A okay so this could be pass tree for this type of declaration and now when I am looking at this particular note I want to associate this type here now how do I carry this type here I need to do certain computation could be a simple copy or it could be certain computation this note, this note, and this note these are the three notes when I encounter one to make can please in the simple table corresponding with very, very saying that type of this is three okay.

And another example we have was that if we have let us say some expression where I want to associate again in certain five values within it so if I have expression like this where I say that E goes to E + E and this is my type expression and let us say that ID 1 and ID 2 have already been declared so when they declared the value use the pipe information is available in the symbol tip, so now what is the type of E there are one to do that computation okay.

Because I do not know whether this is going to be a valid type so remember that somewhere I have gave an example where we have said that A is a sign B + saying or 2 and this one was i and this was string okay normal to find out whether this is valid assignment or not so first I must know for a valid assignment both the times must be same or I should be able to automatically try profit but if the types have to be same then type of B + 2 must be same as int.

What is the assignment to be well now how do I know type of B + 2 so I know type of B from the simple table and type of 2 okay then I have to do this computation to say that the type of this , this particular off set may not be allowed on this kind of attributes okay so this is how you are going to sort of being the types system where I am saying that now I am associating the value where I am saying that I mean I have a type information with it similarly ID 2 will have a type and then this type of information will be carried up in the tree.

Similarly I may say that right information here will be carried at last notes I may be carried from one part tree to the another part or it may carry from the child notes to parent notes so when the attributes are going to become rooted by the symmetric rules and now you can see the symmetric rules for this and symmetric rules for this are going to be different you have to design the symmetric rules okay and their two notations which of them are used one is called synthetic goods another is called inheritor goods.

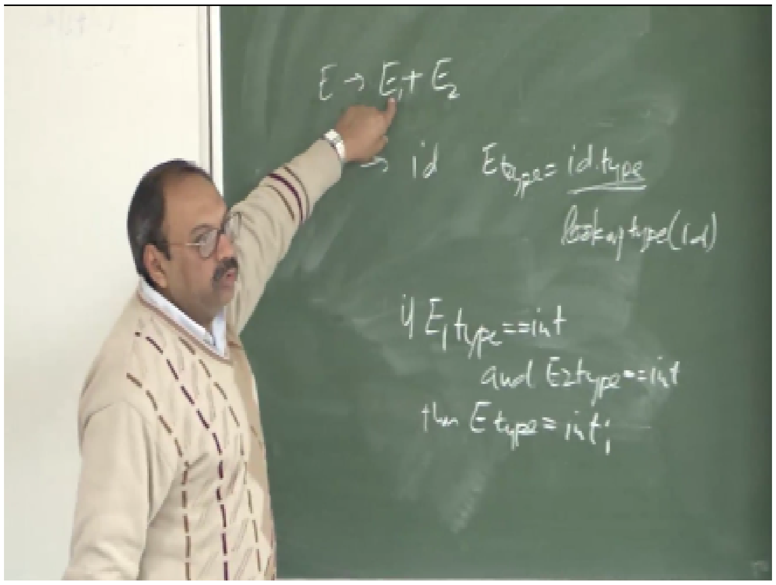
So synthetic goods is the names in S in the situation where do you say that attribute of the note in the pass tree depends upon the attributes of it is children if I know attribute of the children note then I know this computation that is all I need in inheritor attribute is the situation where we said that a attribute defines upon the either the attribute of the parent or the attribute of the siblings okay.

In this case for example I must say that attribute of this part of the tree depends upon the attribute of this part of the tree which happens to the sibling here okay and the attribute of this will depend upon the attribute of this which is coming from the parent so we have two king of attribute associated synthesized attribute and inherit attributes and also we have two notations associated with us.

One is going to be your specification which we are going to call as syntax directed definitions and shortly I will show you what these specifications are so these are only specifications and do not give you any digits about often implement so just say that I have some specification and then I want to implement it an implementation is going to be slightly different what is this we have something known as translation we have obviously we give specifications but we also give an evaluation order.

So for example I can say here that in this case if I say I have an aptitude type then I can always state so let me now put some symbols on this and now when I say E 1 and E 2 I am not saying either two different on the winners I am saying these are two different occurrences of the same non terminals non terminal is E so I realize the rule like saying that whenever I do on this path so whenever I say that our rule which says equals to I D.

(Refer Slide Time: 08:46)



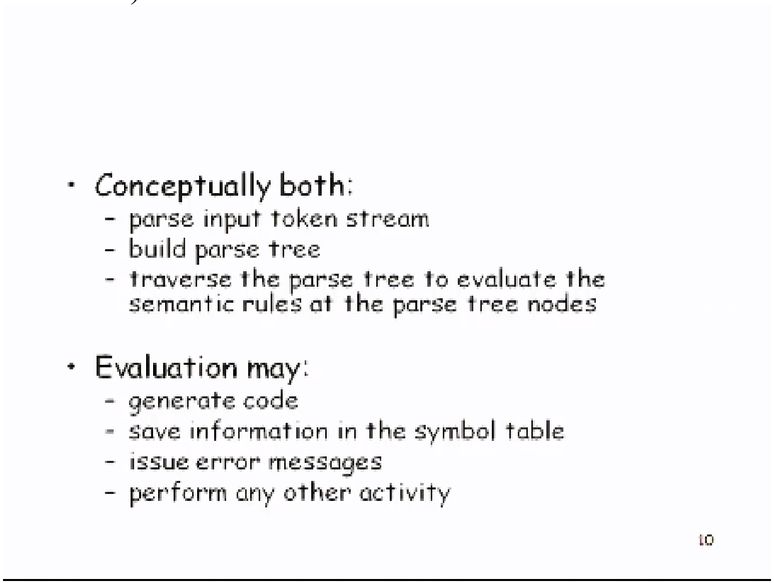
I have an activity equation which says if E type is nothing but ID type okay and how do I know the I D type in the declaration so here actually ID type is nothing but a I will say lookup type information for ID this is what ID type is because ID have already been declared the five information have already entered in the simple table and when I am trying to pass this expression that that is a e-type is nothing but look of this okay.

And then if I write something like this which is says E goes to E1 + E or E goes to E1 + E2 now the reason I am writing one and two are so that I can say that I am talking about an attribute of this, this and this these are three occurrences of the same symbol why I say that I mean I some complex rule by saying that if E1 type is int and E2 type is int then E type is int okay like this so this is saying that if both the sides are integer then this is integer type okay.

And then I can enumerate it saying what are the possible combinations and then I may also have a situation where I say that she type this an incorrect type it is an erroneous type so for example if I say E1 type is string and E 2 type is const then I say that it is an error then I say it is an error so that is how I will be able to flag error seen my crisis so these are really active equations I am writing now this does not tell you if I look at these kind of specifications this is not telling you in which order I am doing evaluation.

That is just saying that I have certain specifications and at some point of time we will have to worry about evaluation of this so we will convert those into translations ints and then use a parser who not only verify that your grammar is correct but also to see that your eye socket same method can be used for code generations bring me to make sense and I will give you example again from yet in the previous class showing that fully return

this okay so if I go back to yet again you have certain dollar symbols associated with which are the grammar symbols here.
If I have a dollar symbols what are the dollar symbols associated with these so I have dollar, dollar with this dollar1 dollar 2 and dollar 3 with this okay I can always and what are the dollar symbols and what is the type of this what are the type of dollar no have you seen the code of Y dot F dot C this is the structure what is the type of structure yes is a union so you can like a 20, 20 1 this cannot say two structures good.
So the next time ask the questions that time DNS right just do at the top so it is actually a union yes you and I ask to anything you recall now so sometime when you try to associate string figure values with this it actually gives you warning saying that you are associating your wrong type unless you bypass first dollar, dollar you eliminate those warnings you see it is not a error but you must know internal of your implementations so conceptually both chart really the same okay.
(Refer Slide Time: 12:53)



- Conceptually both:
  - parse input token stream
  - build parse tree
  - traverse the parse tree to evaluate the semantic rules at the parse tree nodes
- Evaluation may:
  - generate code
  - save information in the symbol table
  - issue error messages
  - perform any other activity

10

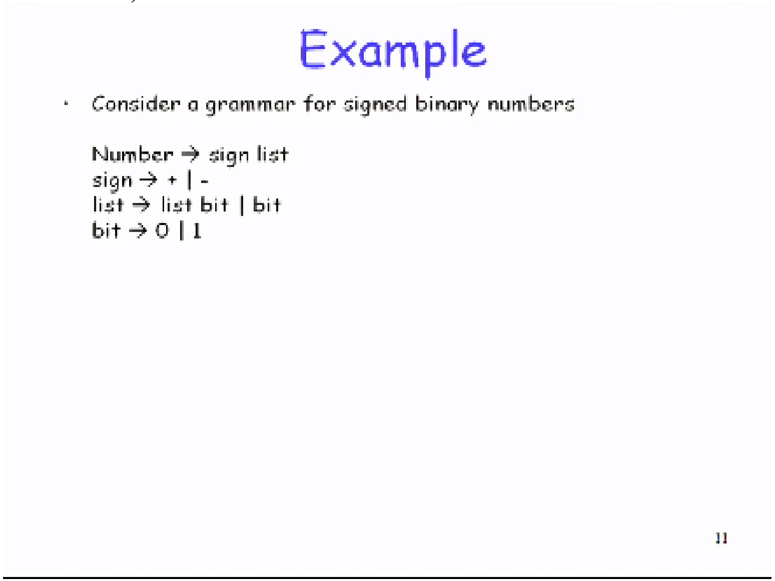
Only implementation are different okay but the intention is that as I am so conceptually one way to look at this is that first I want to parse my expression or parse my programs and once I have constructed the parse tree now that was the travel to the parse tree compute information so for example here I will say that my parse tree is going to be something like and once I am greeted with parse tree then I say that if I now reverse it and does not matter in which order I can let us say traverse it at any order and multiple times then I say that if I know type of this if I have a type of this what is the rule of computation or type of this.

So I have these kind of rules associated it is not giving me any evaluation over it is not giving me any order in which I want to translate but ultimately the goal is going to be that ideally if I can construct parse tree and at the same time I keep on computing my attributes that is same in multiple passes so I try to write my attributes in a fashion that I can do it as I am parsing and then we should be able to see that that immediately leads to translation scheme because then I am saying that this is the order in which I build my parsley and therefore the order in which I to compute my activeness okay.

So we want to now traverse and parse tree reevaluate the semantic rules but please also remember that, that is only a widget is possible in some situations that I will not be able to do it in and I am computing my parsley as I am building my parsing it is also possible that I may have to have multiple passes over this parse tree before I mean in this computation so all those situations are possible so evaluation of these attribute equations it can either generate code.

It can save information in the symbol table so for example if I am parsing this I am trying to save information in the symbol table if I am parsing this and I find the types are not same so I may show an error message in I can perform any other activity so basically what we are talking about is now a famous of top of the consist grammar this is going to be used for implementation of rest of the field of comparisation any questions okay so let us move on so here is a small example so what I am showing you is a grammar

(Refer Slide Time: 15:21)



Example

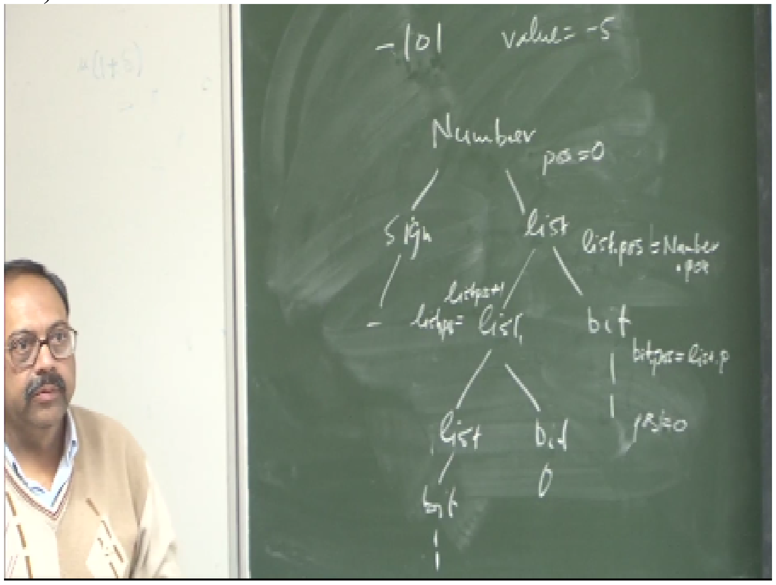
- Consider a grammar for signed binary numbers

Number → sign list
sign → + | -
list → list bit | bit
bit → 0 | 1

11

What is a grammar it is only saying that have a number and the number is can be a sign number and sign could be + or minus anything else and followed by the list and the list is nothing but a list of 0 symbols and binary number either + or minus okay what do I want to do I want to now write some attribute grammar which going to annotate number with value it represents so for example if I write something like if I say the number is minus.

(Refer Slide Time: 15:49)

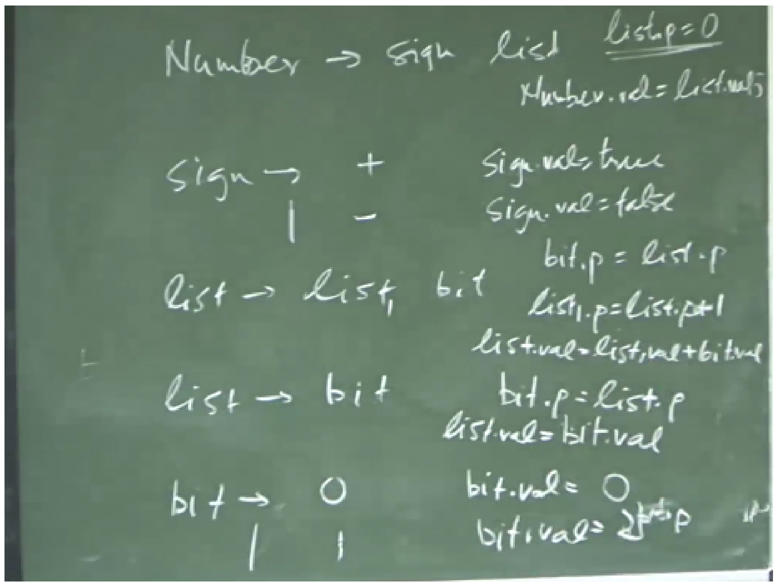


Minus one zero one okay if I pass this using the grammar here I will get the pass tree like this I will say number is nothing got a sign and list and sign is going to be minus and list is you can see rule and this will give me list as list forward by a bit and this will be warm and this will be again a list forward by the bit this will be 0 and this list will give me a bit which will be debug this is how the pass tree going through okay.

And now what I want to do is however associate value saying if I look at this particular note then what is the value of the tree below it okay and then finally we have to find the value of this particular number let us say in decimal okay now if I ask you to convert this binary to decimal how will you do it what is the eclogue to use place value right then this is right source for the place value associated with 22 power 0 22 power 1 22 power 2 an then the end of the will.

So I must know pollution of each bill before I compute this okay so same thing I must compute here I must know pollution of each bit okay similarly to that so now let us see that can I send this rules such that somehow given this past tree and given certain rules which are associated with the each of the notes I can find out the decimal value of this the indent Is clear for the spot ton time to do then so if I say that with number I have let us say tribulate associate value okay.

And finally I want to say value is minus 5 and what it is 22 power 01 and 4 yeah so this on now tell me what are the rules I should write so that I could do this computation that when I finish this when I do this when I even ate all my rules the value of this particular very variable will be minus 1 so think about it so let me what is the position of this bitch this position zero now how do I get this position zero.
So if I say that to begin with whenever I am parsing okay and then I said that this point of time I do not care about the order of evaluation I just want to rise you at the good equations okay I can say that whenever I start creating this tree and it does not matter now whether I am creating it top down or bottom up because I can have multiple passes over it that as soon as I start reading this number I say that rest position let us say position is defined that is so position and now a tribute which is initialized position 0.
And if I say that I have just activated number which is position then I can say that list is a position so list position is so the way I am going to do it now I will say that for attribute of this name list I will have the same grammar symbol followed by the attribute name because I can have multiple attributes so I will now say that list position is nothing but let us say number position so this will gets me value of this value which is zero now I can say bit position is bit position is let us say list position that will also say zero.
And then once I know the position what will I do then I have standard technique I will be file with power of 2 then 2 to the power of position but here I must take this position as 1 and here this position should be 2 right so now I can route write the rule here which says that when I am looking at this list position I can say this position is nothing but so let me call this attribute and this attribute in different names.
So let me call it list 1 so that I can say now list 1 position is nothing but list position + one and then if I now look at this so bit position is nothing but list position so it will get value 1 here and what about this again the same rule will apply here which will take no value 2 and then I can say that this bit takes value 2 right by starting this I can write rules now you can see that if I take this pass tree or this pass tree as long as my grammar rules is the same my acoustic question is this.
Some people are nodding their head but I can see lot of blank faces also are you understanding am I what I am doing here or not somebody who does not understand what I have done this so let me write the types of difference let me type this grammar okay (Refer Slide Time: 21:47)


Number → sign list list.p = 0
Number.val = list.val
sign → + sign.val = true
| − sign.val = false
list → $list_1$ bit bit.p = list.p
$list_1$.p = list.p+1
list.val = $list_1$.val + bit.val
list → bit bit.p = list.p
list.val = bit.val
bit → 0 bit.val = 0
| 1 bit.val = $2^{bit.p}$

So let me write this grammar okay and here I am saying a number is sign list and sign is + or – and list is list forward by bit or list is followed by bit and I say bit is either 0 or 1 this is what my grammar is okay now suppose I say that so let us not worry about sign for time being okay sign I will see later suppose I sat to begin with here I say that now I am writing certain rules in this so this is my grammar and I am going to add rules to this so if I say my rule is something like this say that list p is let us say initial as to 0 or I can say it is number . position that is the rule associated with this particular grammar rule .

So this is an attribute equation and then I say when it comes to this grammar rule I say that the bit position in that so let me differentiate between this list and this list and I am saying this as list 1okay so now I say that bit position is list position and list 1 position is list position +1 this is the rule I have applied okay and here I say that bit position is nothing but list position and then is say that bit value then I have to compute the value what is the bit value here 0 will always give me 0 does not matter where it occurs so the position is not important and so I can say that the bit value is 0 and here the bit value is $2^{\text{bit position}}$.

So this is now telling me that the level when I can look at the contribution of the bits if it is 0 it does not contribute anything okay but then I need to know what is the value of the number so I need to take the information that is if I say that what is the list value is nothing but the bit value whatever is the value of this bit that is which is going to be copied here okay so this is bit value so what about this when I have a rule like this what is the value of the list now to compute this value .

I must know this value and this value right and I just need to add the two so now I can say that list value is nothing but list 1.value + bit value right and if I want to find out what is the number value and the other side let us ignore for the time being okay what is the number value it is nothing but the list value right you can see what are our specifications it is telling me that how do I compute this value of the number okay I do not know which order we do the computation I will find it out okay.

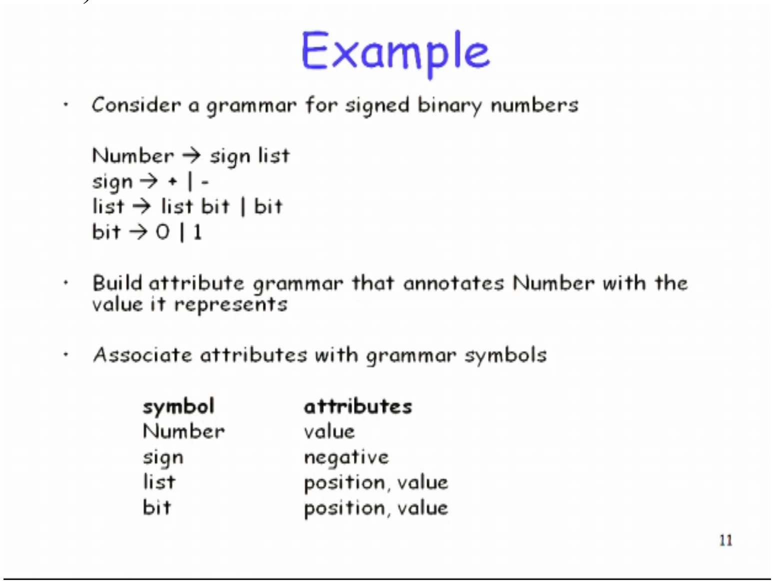
Now let us do this suppose I am now worried about this sign okay so now I can say that in this case so let me write now slightly differently so I have two signs okay so now I can say that this is either it is positive or negative okay so I can just use a boolean here and I can say that sign value is true and this is saying that sign value is false okay so I can use any attributes and now this rule is going to change slightly and now I will say that the number value .

Now I can say if sign value is true then what is the num value then I say the num value is let us say sign list value else minus list value right so I know that if ever create a parse tree using these grammar rules and then I evaluate all these rules and parse tree I will get the number and on the other side at this point of time we have specifications I do not care about in which order I am evaluating these specifications that we are going to do it at some later point of time.

So list is the non terminal I have and list 1 is just another occurrence of the same thing I want to differentiate between the attribute of this list and this list now when it come to implementation I already have a method which says that this attribute is $ and this attribute $1 okay so I will be able to differentiate when notation ally I use here I need some so I am using subscript but this not a new non terminal this is the same non terminal .

So now if I use all these rules on this parse tree you can see that in one top down parse I am going to get all the bit value and bit positions.

(Refer Slide Time: 28:53)



Example

- Consider a grammar for signed binary numbers

  Number → sign list
  sign → + | -
  list → list bit | bit
  bit → 0 | 1

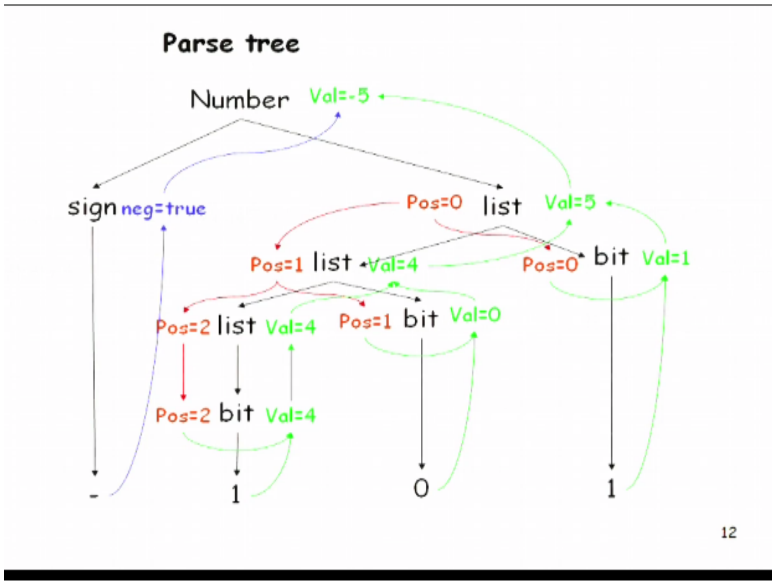
- Build attribute grammar that annotates Number with the value it represents

- Associate attributes with grammar symbols

symbol	**attributes**
Number	value
sign	negative
list	position, value
bit	position, value

11

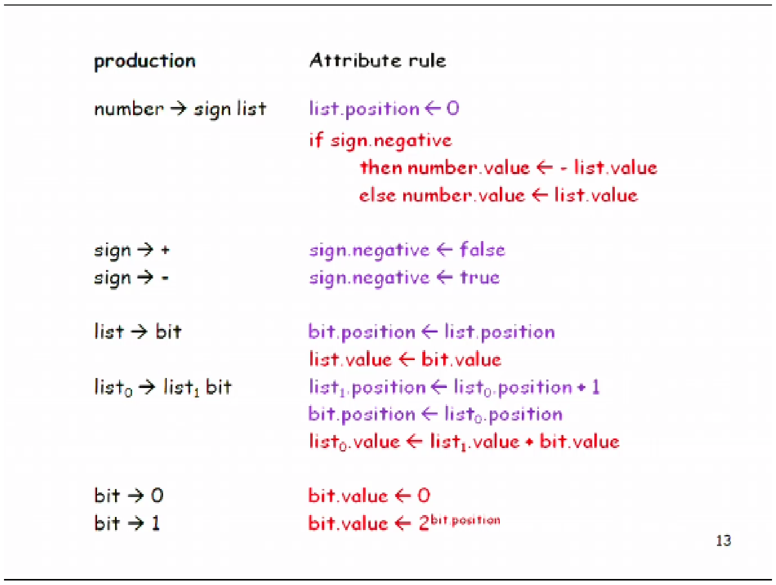
(Refer Slide Time: 29:54)



Now this is how the parse tree will look if I put rule s in this so the value is negative instead of these numbers I could have so this is a general matter which say that if I have certain property with these loads then how do i and what I am using her I am using a combination of in some cases you see that this value whatever values I have put in green actually synthesizes attribute because it is defined in terms of children nodes and whatever values you see in red is actually coming from the parent and these and you can inherit certain properties or you can synthesize certain properties okay.

Now so there are 2 things one Is the parse tree you see in black color and then you have all the blue ,green and red which I the dependence graph which is saying that this evaluation change certain order. Now the orders of construction of the parse tree and evaluation of all these attribute and because I am writing only certain properties not worried about the evaluation but at some point of time to worry about this okay.

(Refer Slide Time: 32:19)



production	Attribute rule

number → sign list	list.position ← 0
	if sign.negative
	then number.value ← - list.value
	else number.value ← list.value

sign → +	sign.negative ← false
sign → -	sign.negative ← true

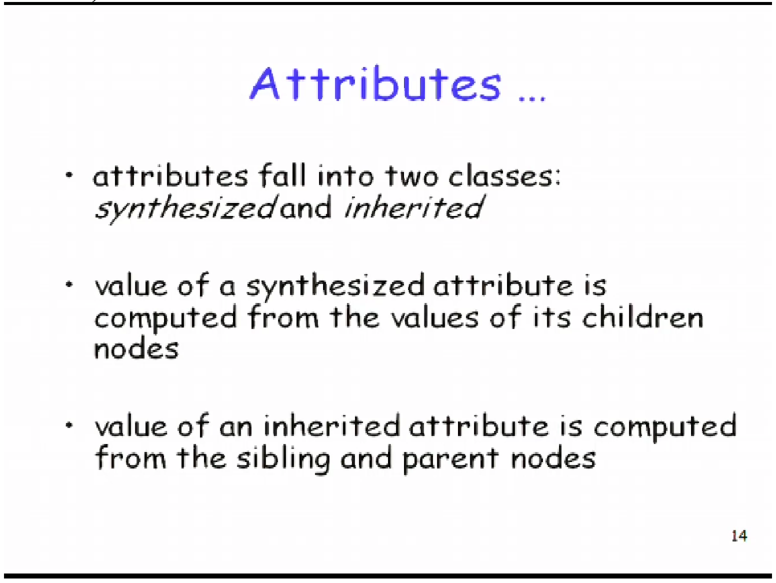
list → bit	bit.position ← list.position
	list.value ← bit.value

$list_0$ → $list_1$ bit	$list_1$.position ← $list_0$.position + 1 bit.position ← $list_0$.position $list_0$.value ← $list_1$.value + bit.value
bit → 0	bit.value ← 0
bit → 1	bit.value ← $2^{bit\ position}$

13

This is what my grammar this is how rules are going to be the list position is 0 so this is just a replication of that this say that sign negative is false and this ay that sign negative is true and the bit position is nothing but the list position and list $_0$ position is list $_1$poition and the 0 is bit value and the bit value is 2 $^{\text{bit position}}$ and then I start taking these values up and the bit value is nothing but the list value and then if the sign is negative then the list value is negative then it becomes number value and I have some of the attribute grammar and evaluation of attributes.

(Refer Slide Time: 33:25)



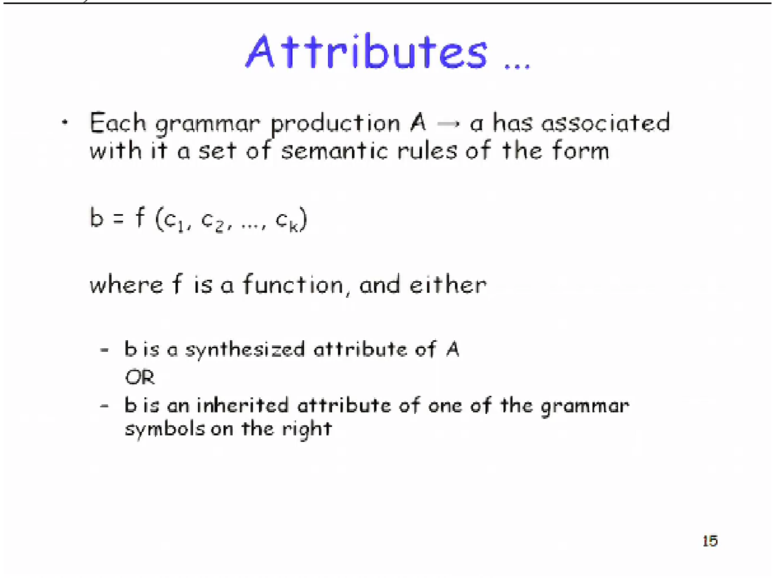
Attributes …

- attributes fall into two classes: *synthesized* and *inherited*
- value of a synthesized attribute is computed from the values of its children nodes
- value of an inherited attribute is computed from the sibling and parent nodes

14

Now let us take another example attributes. I have already discussed attributes fall into two classes one is synthesized attribute and another one is inherited attribute and the synthesized attribute is computed from the values of the children node and inherited attribute is computed in terms of the attributes of the sibling and the parent nodes and having something the standard English words okay.

(Refer Slide Time: 33:45)



Attributes ...

- Each grammar production $A \rightarrow a$ has associated with it a set of semantic rules of the form

  $b = f(c_1, c_2, \ldots, c_k)$

  where f is a function, and either

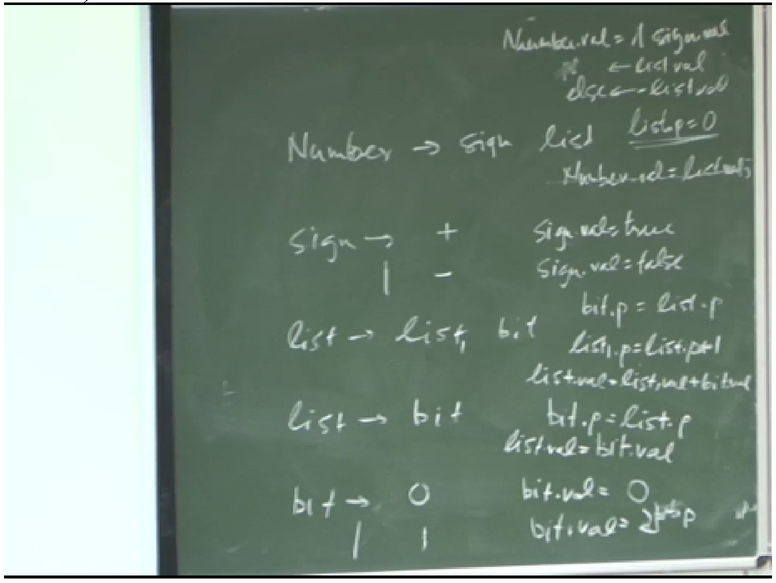
  - b is a synthesized attribute of A

    OR
  - b is an inherited attribute of one of the grammar symbols on the right

15

So we will start putting this in notation so what we are saying now is suppose I have a production of the sentence form A a and then we can associate it with a set of semantics I need not have one rule so you can see that for example when this particular grammar rule I have two rules associated which says list position is bit position and list value + bit value this values are to be associated with the same grammar rule.

(Refer Slide Time: 34:23)



Number.val = if sign.val
← list.val
else ← -list.val

Number → sign list   list.p = 0
Number.val = list.val

sign → +   sign.val = true
| −   sign.val = false

list → $list_1$ bit   bit.p = list.p
$list_1$.p = list.p+1
list.val = $list_1$.val + bit.val

list → bit   bit.p = list.p
list.val = bit.val

bit → 0   bit.val = 0
| 1   bit.val = $2^{bit.p}$

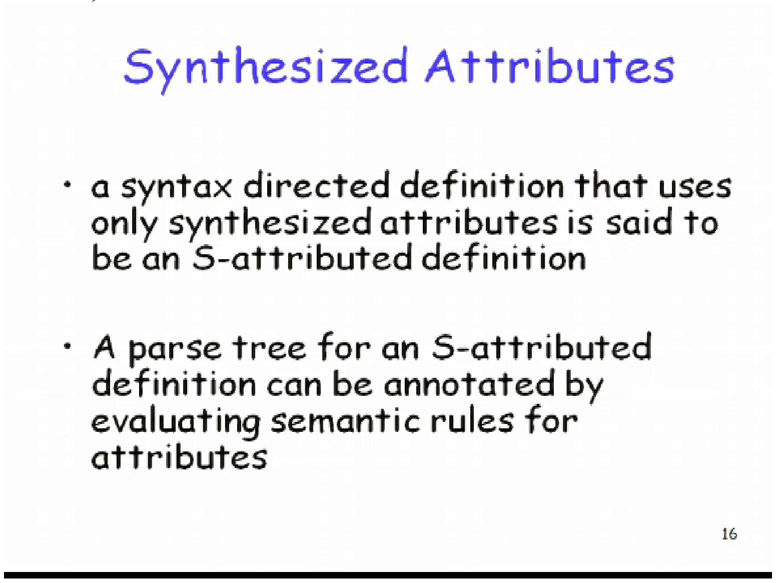
So in general I can say that m rule is going to be of this form where I say I am trying to compute b which is a function of attributes c_1 ,c_2 and I do not consider whether it is synthesized attribute or inherited so if it is synthesized attribute then c_1 to c_k are attributes of the children or if this is inherited attribute of the grammar symbol on the right hand side then what it means is that c_1 to c_k are attributes of parent node and siblings.

So attributes what we have to remember here is that this attribute depends upon attribute c_1 to c_k so I cannot compute b before I have computed c_1 to c_k so looking like this I for example you cannot compute bit position without knowing the disposition I cannot compute list solution without doing the list position and the value and only computes my attributes in the order that no of none of the edges get and I also assume at this point of time that we do not have sizes so that means like then what is going to change the property of the dependence graph .

You have can you for example on a dependence graph which does cycle you are sorting on that and you sort that graph and find out the variation order not give your total order but we give you partial order all right so if I do a topological sort on that then I know what my partial order of the variations right so for example again once going back to that figure once again so if I know that I can compute these in certain order then you know that if every time I am computing where all the things have been computed then i have no problem the order is not important .

So for example I compute this value first or this value first it is not important as long as this computation is considered and if e goes to e1+e2 does not matter whether I compute e1 first or e2 first as long as I have computed of course at some point of time impose certain constraints and continue on that so we have two kinds of attributes so we are going to look at the synthesized attributes.

(Refer Slide Time: 37:29)



Synthesized Attributes

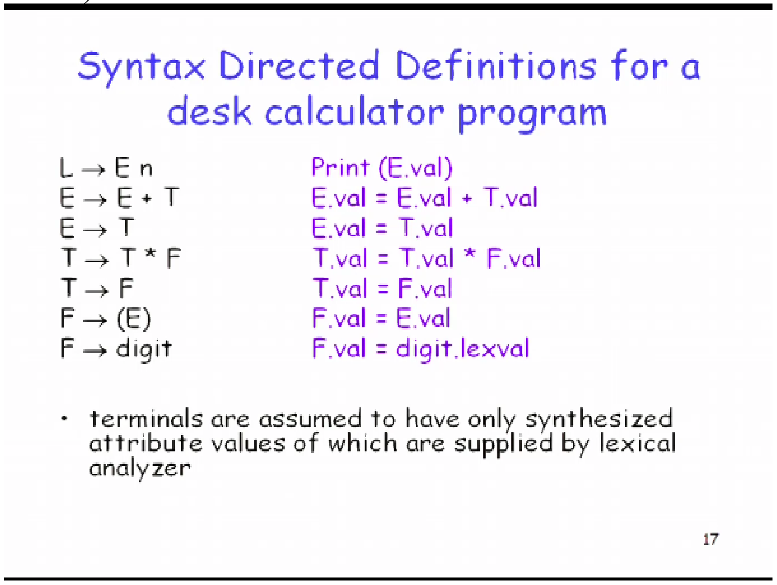
- a syntax directed definition that uses only synthesized attributes is said to be an S-attributed definition
- A parse tree for an S-attributed definition can be annotated by evaluating semantic rules for attributes

16

So synthesized attributes are actually a syntax directed that uses only synthesized attributes is said to be an S –attributed definition and a parse tree can be annotated b just looking at the semantic rules and then I can say for each node I can allocate it with the semantic rules so what will be the parsing method for construction of dependence graph or synthesized attribute the bottom up parser and if I have only synthesized attributes then I know that either I am constructing my node ,the nodes below it have already been constructed.

And I mean keep on computing and finally when I reach the loop node by then I have the computed attribute okay so here is an example which is coming from the desk calculators program .

(Refer Slide Time: 38:20)



Syntax Directed Definitions for a desk calculator program

$L \rightarrow E\ n$	Print (E.val)
$E \rightarrow E + T$	E.val = E.val + T.val
$E \rightarrow T$	E.val = T.val
$T \rightarrow T * F$	T.val = T.val * F.val
$T \rightarrow F$	T.val = F.val
$F \rightarrow (E)$	F.val = E.val
$F \rightarrow$ digit	F.val = digit.lexval

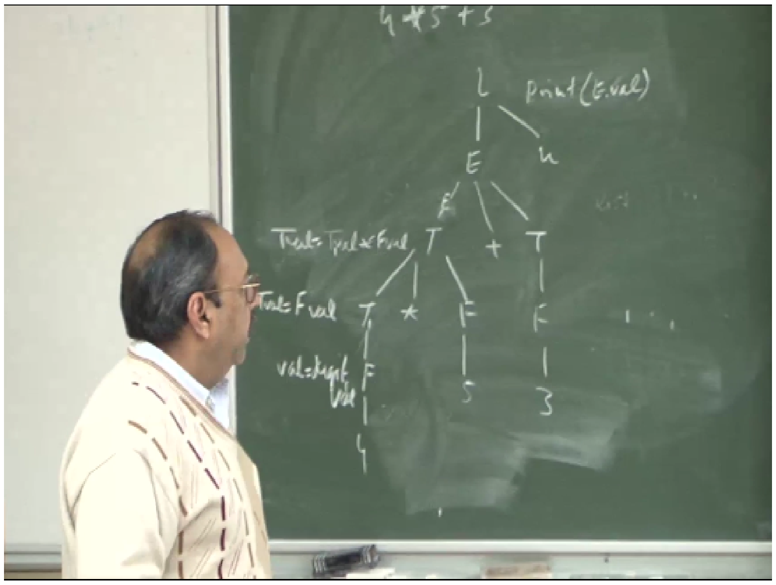
- terminals are assumed to have only synthesized attribute values of which are supplied by lexical analyzer

17

This is actually a program in C you can use binary calculator but then they were also calculators like here I could type an expression and finish typing an expression it gives me the value of this code right this is I mean if you take any calculator what do you type you are typing an infix expression right and when you press return gives you the value of that expression means it works internally what it is doing is nothing but parsing your input suppose this is the grammar.

I am using for parsing my input it says that when I press return okay so n is saying new line or return then L gives me the value of this expression and my expression grammar still remains still e goes o t and e goes to t t goes to t*f and t goes to f and f goes to digit expression the rules . I am going to write so suppose I say that I just want to make sure that when I finish typing my expression invest return value of this expression is printed on the screen so as I see digits and I as I see operators I must be able to compute all the values which are associated with this expression so for example if I leave now some expression which is passed by this particular language.

(Refer Slide Time: 39:49)

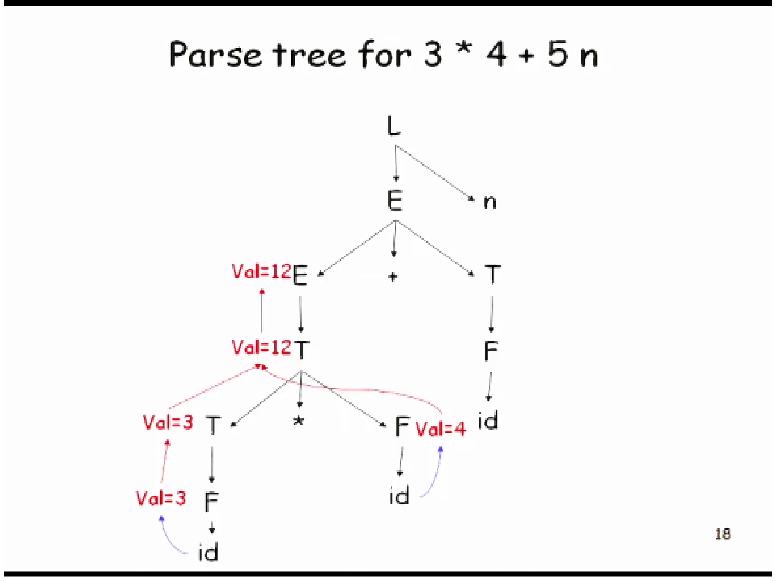


Let us say that I am trying to compute and say 4*5+3 some random number what is the kind of parse tree I will construct for this an expression so you say that L is followed by an expression followed by u line and then I want to clearly say that l value here is nothing but e value or I can simply write it in terms of saying that here instead of that I can say print you I can just say that print but if I look at this part and now you can see that what kind of parse tree I will get from this it will give me t star F then p star f actually get reduce to E and t will get reduced to e so this is how it will come .

So idea I am assuming our numbers say this part of the parse tree will look something like this and this part of the parse tree will look e is going to be so he and actually I should have had one more e here logically e + t and this t is going to f and now if I write all these rules what is if I say my value or attribute that is associated with is just the value then what will I say what is the value here 4 but I need to write the rule.

I cannot just say that when I reduce by saying F goes to digit so what is the rule I write so I can say that value is nothing but here digit value that becomes my general rule okay so same rule will apply here and here which you say that value of F is nothing but the digit value and when this rule is used which says T goes to F then I will say that what is t.val is nothing but f value right so it is just popping so whenever I do a reduction by this rule it says that e value is nothing but f value and what is the rule I write for this when this is equals to e star F okay. So what is t value is equal to e value * f value only one small problem that I do not know what this t value is because there are two values occurring so I need to make sure that I use some subscripts here to

differentiate between them and therefore I say that this is e 1 and therefore this as t1 value and similarly I can now say that when e goes to t c I will say that is t value and so on okay .
So this way I can very quickly create all my rules and what I am saying here is that f value is nothing but whatever is value of the digit then f.val is e value and t value is f value and this is saying t value is t value + f value and the same e value is e value and terminals which are associated have only synthesized attributes and value which are subscribed by lexical analyzer.
So for example this is a terminal these values are going to come only from the attribute so this is how my parse tree will look.
(Refer Slide Time: 43:38)



And then I can do all the applications of these values okay so I had expression here so basically what has to be clear to you is that given the grammar and given a problem to be solved on not unloading several problems one problem I posed was that given a binary number assign binary number how do I find equivalent decimal number or given an expression how will I find value of the expression of given a type how do I find type of each of the variables these are different problems and for each problem I should know how to write my attribute equations my grammar will remain the same so using the same grammar same parsing technique that you can solve different problems.

Acknowledgment
Ministry of Human Resources & Development
Prof. Phalguni Gupta
Co-ordinator, NPTEL IIT Kanpur
Satyaki Roy
Co Co-ordinator, NPTEL IIT Kanpur
Camera

Ram Chandra
Dilip Tripathi
Padam Shukla
Manoj Shrivastava
Sanjay Mishtra
Editing
Ashish Singh
Badal Pradhan
Tapobrata Das
Shuubham Rawat
Shikha Gupta
Pradeep Kumar
K.K Mishra
Jai Singh
Sweety Kanaujia
Aradhana Singh
Sweta
Preeti Sachan
Ashutosh Gairola
Dilip Katiyar
Ashutosh Kumar
Light& Sound
Sharwan
Hari Ram
Production Crew
Bhadra Rao
Puneet Kumar Bajpai
Priyanka Singh
Office
Lalty Dutta
Ajay Kanaujia
Shivendra Kumar Tiwari
Saurabh Shukla
Direction
Sanjay Pal
Production Manager
Bharat Lals
an IIT Kanpur Production