

**Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 16**

By...

Prof. S. K. Aggarwal.

Dept. of Computer Science and Engineering.

Good morning so let us start our discussion from past class starting looking at how we will study of this topic and one example of a regulation grammar and time to see that all the rules we had

just been used to copy

(Refer Slide Time: 00:40)

stacks

- bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack
- synthesized attributes of Xs can be inherited by Y by using the copy rule $Y.i = X.s$

Example :take string real p,q,r
 $D \rightarrow T$ ($L.in = T.type$)
 L

$T \rightarrow int$ ($T.type = integer$)
 $T \rightarrow real$ ($T.type = real$)

$L \rightarrow$ ($L_i.in = L.in$) L_i .
 id ($addtype(id.entry L.in)$)

$L \rightarrow id$ ($addtype(id.entry L.in)$)

So we can do one different partial p there are could be eliminated by using inspiration of the state and these two actions are initializing attributes to this two actions it is a single table this only one confirm therefore we can eliminate other rules then are the three labels then are three labels to this end of the stack of here.

(Refer Slide Time: 01:06)

State stack	INPUT	PRODUCTION
	real p,q,r	
real	p,q,r	
T	p,q,r	$T \rightarrow \text{real}$
Tp	,q,r	
TL	,q,r	$L \rightarrow \text{id}$
TL,	q,r	
TL,q	,r	
TL	,r	$L \rightarrow L, \text{id}$
TL,	r	
TL,r	-	
TL	-	$L \rightarrow L, \text{id}$
D	-	$D \rightarrow \text{TL}$

Every time a string is reduced to L, T.val is just below it on the stack

And we went to this example of final output whenever certain deduction of taking space that means I have a reduction this is either almost L from ID or L from L, ID I will define value of the stack information above this stack.
(Refer Slide Time: 01:22)

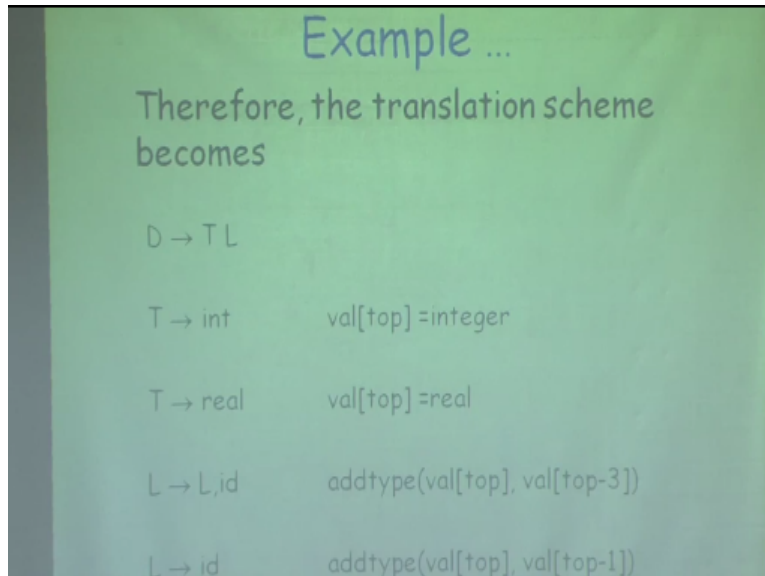
Example ...

- Every time a reduction to L is made value of T type is just below it
- Use the fact that T.val (type information) is at a known place in the stack
- When production $L \rightarrow \text{id}$ is applied, id entry is at the top of the stack and T.type is just below it, therefore,

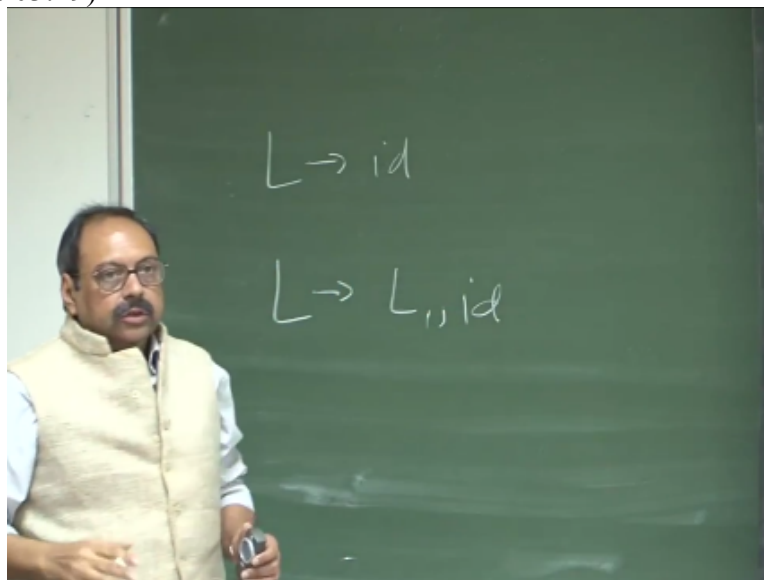
$$\text{addtype}(\text{id entry } L.in) := \text{addtype}(\text{val}[\text{top}] \text{val}[\text{top}-1])$$
- Similarly when production $L \rightarrow L, \text{id}$ is applied id entry is at the top of the stack and T.type is three places below it, therefore,

$$\text{addtype}(\text{id entry } L.in) := \text{addtype}(\text{val}[\text{top}] \text{val}[\text{top}-3])$$

And using this then you found that actually this entry because corresponding to L going to Id in case we finding out that the right information we started this case we are going reduction and improve this sets I will going to L going to L, Id and I had corresponding action here I was able to find that valuable of the type for that deduct of minus T.
(Refer Slide Time:01:48)



And let us identify this factor I was an enable to eliminate of the copy rules and was straight away to put the inputs and this is called initializations and these two T s or these actions for the entries right side quotes right this is what we will doing okay and now let us see what is the property I identified in placing all those actions of copy my state and using the input from okay. So what is something that the they listed some unique property which has should be looking for so that I can always use this string in general that signified that if cannot generalize it then this is the not going to a stack okay so I want to find a general solutions to all the terms so what is the characteristics you know this in this particular grammar so that difference now so two things can let me once again bringing the simple purpose so what are the rules grammar rules.
(Refer Slide Time: 03:19)



I was able to replace all these information by stack information one was that when I hide this reduction which said that L goes to I D and another L goes to L, Id needed two times I was able to do this replacements and in this case ever said Id was always that top minus 1 and in this case type information was always a pop minus T right.

So what is unique about this is something unique so what is rule about this is that does not matter when I am doing reduction if I do this reduction there is a speed coming place in this stack rule where the informations are available similarly when I do reduction by this rules when the speed determine the place when the stack where design to stack coefficient is available so I suppose I get into the situation where I do not know the what is speed coming places then the string going to change okay.

(Refer Slide Time: 04:24)

The slide contains the following text:

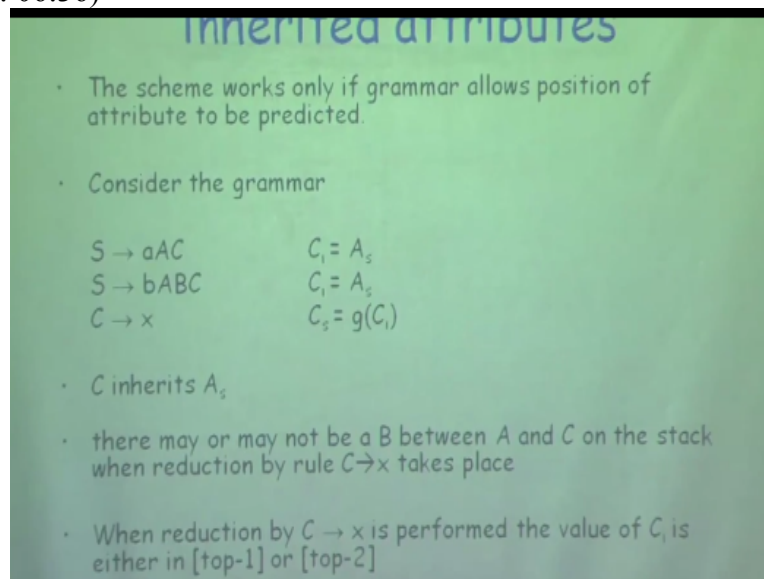
- The scheme works only if grammar allows position of attribute to be predicted.
- Consider the grammar

$S \rightarrow aAC$	$C_i = A_s$
$S \rightarrow bABC$	$C_i = A_s$
$C \rightarrow x$	$C_i = g(C_i)$

So let us look at reduction example of a grammar like this okay so take this grammar we have two for us one S goes to lowercase a Ac and this A that goes to B and ABC and C goes to X okay stack is that C inherited is a copy of S inthesise so behind to this continuity of this C is a is a copy of this syntasize and corresponding to this rules I have C has going to of C has define function of Ci now what happens can I apply scheme okay.

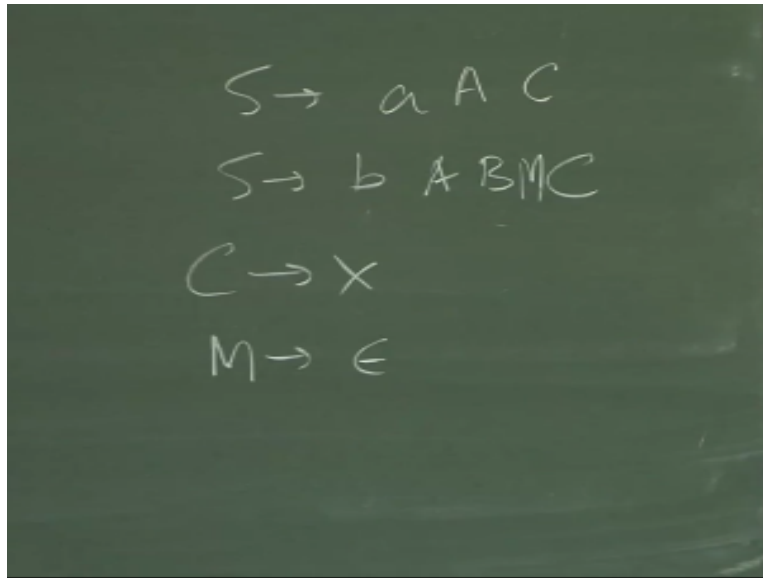
Able to same scheme work that I say that I do not have to worry about finding out the question had this is nothing but copy are the phase in this design therefore if I need to use this rules then there is no need to really copy C is here and that using as in hesitant that possible yes No but you're saying no that quite interesting yes so focus on this particular this is so I am saying that I am using C there so I am saying that instead of making a copy not to use a synthesized in this

emphasize is already on the stack now when I'm doing this reduction when I go deduction of x to c do I know where is the synthesizer is.
 (Refer Slide Time: 06:36)



So if I am doing passing what will be the configure of the stack let me assume two configuration of this stack okay either my these are only the talking about this X stack and the value attack my S stack configuration with either B a capital A X will be the reduction it will be b capital A B and X these are going to be my stack configuration before I do this reductions before I say that this C either goes Xi this Xi goes to C and this time I said that are the compute this synthesizer which happens to be a function of C there which is happens to be a copy of a synthesizers okay. That means either the value of the A synthesizer is here and the corresponding values is stack and the corresponding value stack is contain a new synthesizers problem is that I am doing this reduction okay I do not know which rule I am going to use next for either this reduction of this that one I find a new rule I am talking about general about what are the purpose so when I do this particular reduction that is not clear. Whether next internship is going to be by the first rule or the second and therefore reduction of A when this reductions happens this now could be below that top minus on or top minus that we have the terms what happens and now how can you lose information so that I can always find you that affects place we talk about the general thing what are the general thing what are the rule of reduction to the end top up you can reduce somebody say something a addition of the stack epsilon on the similar.

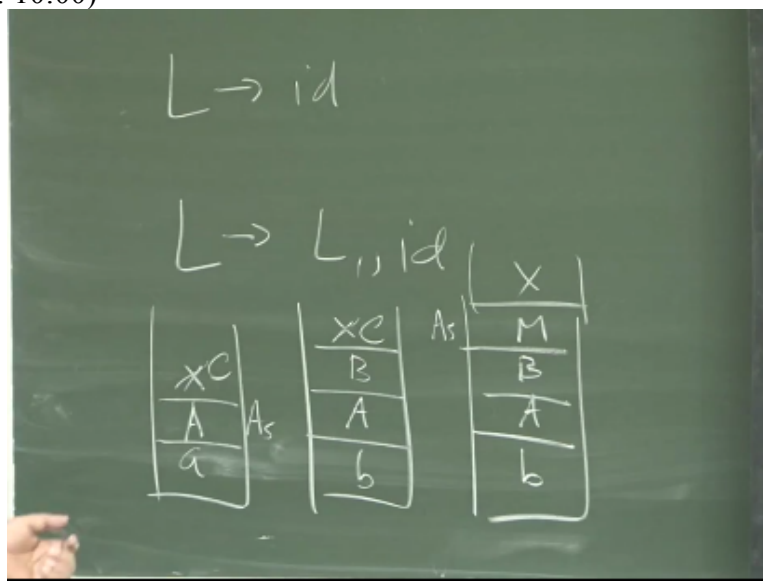
(Refer Slide Time:08:56)



Now suppose I say similar to here and my rules something like this id I say so my directions to the end somebody has to system all these mothers not suppose I do something like this if I say goes to a AC and then I say S goes to b ABC and then I say C has to be X instead of this okay I am not making a small change and saying that let me put a mark from her and then I say M goes to epsilon okay.

Now I can do this that now you can see that if a synthesize C is below this and now this copy of this synthesize then does not matter whenever I do the reduction of the rule A implies this that going to b rules okay then use this step.

(Refer Slide Time: 10:00)



And so now suppose my stack configuration in this case becomes when I am taking my second rule the stack configuration something like this b A B M and X and now this marker is used to

keep your copy of phase in themselves now suppose I make a copy of a synthesized here I made a copy of a synthesized here then is very difference between this reduction in this reduction as for as solve this reduction in this syntax is always below.

So what I can do when I am leading you towards the general solution that in the first case why the scheme works because whenever I did a reduction I was always able to find a location for my in stack hesitation if I now make this similar configuration of the grammar and reduce the grammar you can see the this transformation not will to do the change the language I am trying to pass okay Therefore if I now make this change okay so problem is that decide the top minus or top minus 2 which I does not know.

(Refer Slide Time: 11:05)

Simulating the evaluation ...

- Insert a marker M just before C in the second rule and change rules to

$S \rightarrow aAC$	$C_1 = A_s$
$S \rightarrow bABMC$	$M_1 = A_s; C_1 = M_s$
$C \rightarrow x$	$C_s = g(C_1)$
$M \rightarrow \epsilon$	$M_s = M_1$

- When production $M \rightarrow \epsilon$ is applied we have $M_s = M_1 = A_s$
- Therefore value of C_i is always at [top-1]

And say that I am introducing a marker to certain rules and the rules set up that M inherited is now I copy of your phases in this and then I say C synthesized is nothing but a copy of N and synthesized and then when I do this particular reduction I am saying that synthesized attribute will be nothing but a copy of the rule and then I can always say that C is a function of c synthesis now I know that I am talking of C in hesitant.

In which is a copy of a synthesized this M_s is also you can see the of copy of n synthesis right so this rule so a synthesized is copied into M_i is copying into m_s is used just here just below it so in this case is that I will have a sailor in this case I will have the S_i .

So I am not have to worry about which rule I am going to use for reduction next I want to say that whenever you reduce by this would which says C goes to X always finds the attribute.

Just grow it okay that still has a limitation so now whenever we apply this production I can always find that and synthesize nothing obvious in synthesize therefore value of C which is nothing but of you can A synthesizes all to the top minus 1 so I want that to do again when there

is still a limitation in what is that limitation in fact there are still not solved that as I said I'm leaning towards the general solution but I have still not reached positive limitation. So where did the scheme failed so first problem I started with last I am saying that I will always have attributes at a predetermined location in the exception I am make the all that most of the time as are nothing but copies of the scheme okay which is the failure if it is not a copy of just another attribute so what happens now if I say that C_i is if you say C_i is the copy of S and M is the copy of S suppose C_i is not a copy of a C and C_i is a function of X .
 (Refer Slide Time: 13:46)

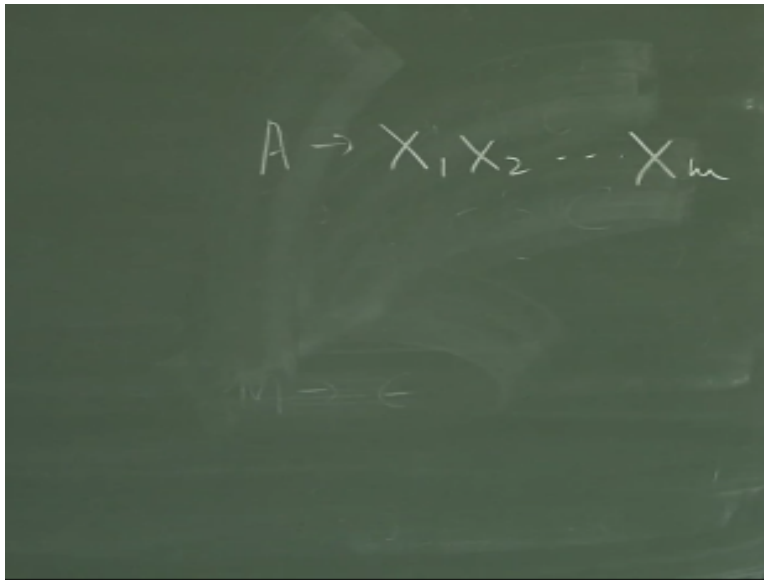
Simulating the evaluation ...

- Markers can also be used to simulate rules that are not copy rules

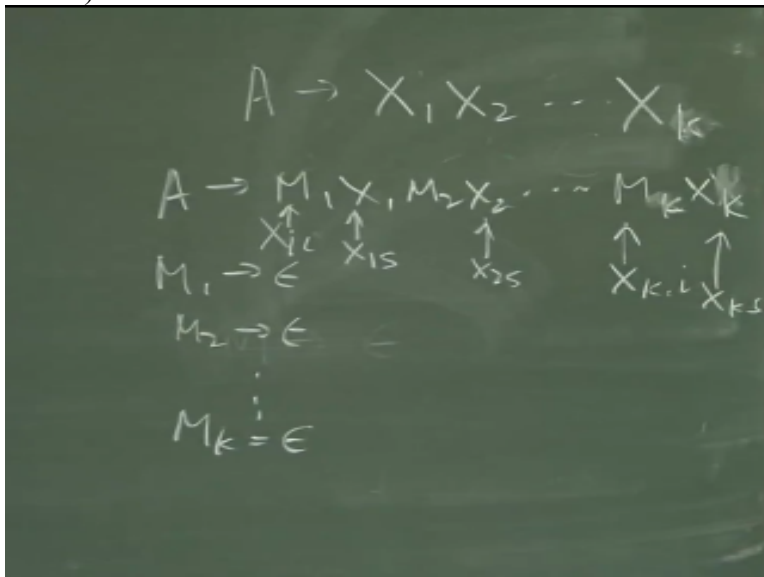
$$S \rightarrow aAC \quad C_i = f(A.s)$$

- using a marker

So let us look at this one now when I say this is not a copy and this is a function I need to be evaluate and again I am going to use same general thing that I am going to put all this marker and the marker is the place will bolder where the attitude the but after the reevaluate okay now this gives me the most general case where I say that does not matter whether you are a evaluate is a copy or a function of I can do and I can always now. The T value and then keep it on this stack in the evaluation position okay now with this I have all these rules I have not enough machinery available to me okay so now let us move towards generalization of algorithm and this point of time what I am going to propose this and let us see that I have in general rule.
 (Refer Slide Time:14:43)



Which is A goes to X1, X2 up to X m okay and I can have attributes which are the copy right that can have a rule which on not copy in the features and I can general rules X okay so what are the can I do now this corresponding to every symbol on the on the right hand side I can lightly okay and if I use a marker then what will happen my grammar will become something like this. (Refer Slide Time:15:22)



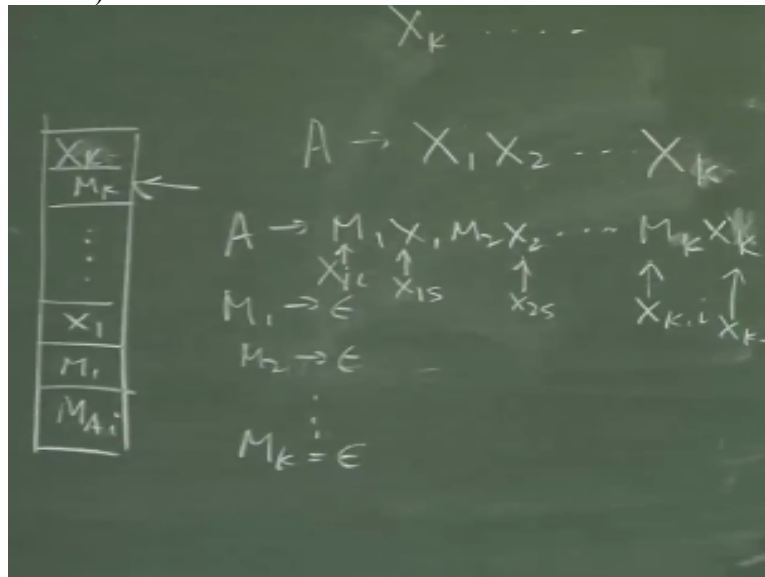
This will be the one change I will do and then I say that all these ends so and one goes to epsilon and M2 goes to epsilon and Mk goes to epsilon and what are these marker hold now what they can hold now is remember that now go back when I was talking about evaluation of this and if you have a proper evaluation order that we have the attributes of symbol. Must evaluate in the position which is immediately that will okay that means I can now say that m1 actually contains X 1i and K contains X Ki right so all marker symbols are try to contain the

behind that you and all the symbols themselves on the value step obviously and all the symbols grammar symbols on the value stack they are going to contain the synthesizer so now I can say that corresponding to this location X is synthesized.

Corresponding to this location I had a S to synthesize I have S k synthesize now we will see the value stack all the behind attitudes then be there I n all etherizers okay and depending upon when I have says this weather this is a copy or this a function and going to have the appropriate the right hand side it is just a copy then I copy it from the coming location and if it is function and there is a function reduction now I have the pre determine reduction for everything.

So suppose now I'm saying that a reduction happens to let us say M_k now when I say reduction happens to M_k what is attributes around compute I am computing M_k inherit the rule which basically is expected to compute X_k inherited and place here what do I need to know I need to know all these attributes so in heritage attributes of A right this a inherited attributes definition oaky away we will be the inheritate attributes of A on the stack there will be find a inheritate attributes of A on this when we have a on the location we have that A is an left hand side hand side in the production corresponding to that will be marker.

(Refer Slide Time: 18:11)



So if I start now showing you the stack this is how the stack we look okay that I will have corresponding let's say marker of A should contain value of A and then I will have M_1 , X_1 of 2 and M_k and X_k now suppose I'm doing reduction to this X_k as corresponding and that the reductions behind the x_k that is going to be her and now I can find and everything it depends upon the steps these are determined these computed to inheriting.

And if I want to if reduction is taking place to let us say X K which is synthesized attribute okay then what are the does not upon which is reduction happens then all the attributes of the right hand side corresponding to this rule which said XK is on the right hand side must have in on the stack so before I pushed X k an X k I am going to that compute the synthesizer attributes form pop on the symbols and push X k push the synthesizer rule of inherited.

We used that will reduce okay but this tell us scheme seems to be slightly flawed I am using the rule of this space right so how can I reduce space does not have this markers are occupying this very good so markers are occupying then what we will we do so if markers symbol does not interstate attribute then we will be only the marker so I can throw away the some of the markers I will only the markers only have in heritage.

And now that is still because I know the markers are throwing away okay that only needs the re computation of the off sets but this will always be the pre determine rules right I can reduce number of markers in that certain hierarchical that rules that I will not require almost certain symbols I do not have inheriting attributes therefore I will not require and could I have been this rules that not try I could now be quit.

(Refer Slide Time: 20:25)

General algorithm

- **Algorithm:** Bottom up parsing and translation with inherited attributes
- **Input:** L attributed definitions
- **Output:** A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \rightarrow X_1 \dots X_n$ introduce n markers $M_1 \dots M_n$ and replace the production by

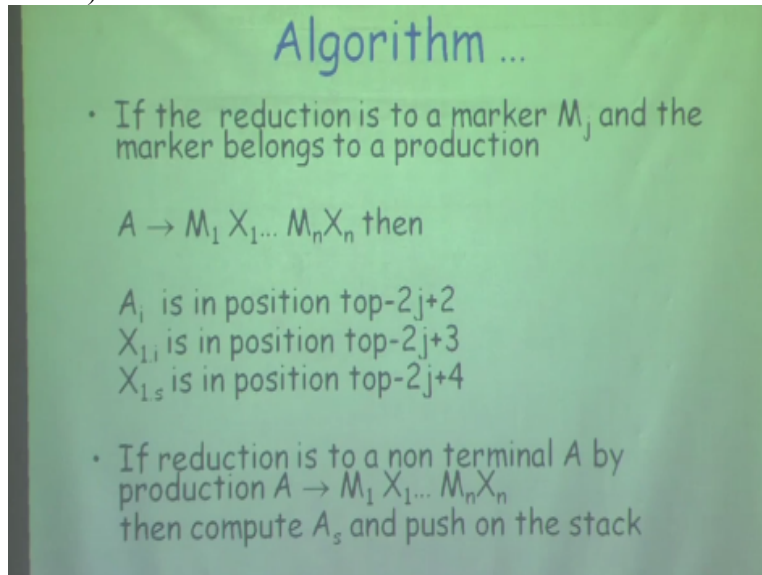
$$A \rightarrow M_1 X_1 \dots M_n X_n$$

$$M_1 \dots M_n \rightarrow \epsilon$$
- Synthesized attribute X_{i_s} goes into the value entry of X_i

So in general what I can I do this that I am using for that bottom of a parser and the parser and translation to lose my input are L attribute of definitions and the advantage is about a parser so I use this form every non terminal which has one in heritage that we can do that we the grammar symbol have does not have certain symbols I do not have in habitant attributes required in general what can I do form which this A goes to X1, X2, X n and I am going to introduce.

The markers and one inheriting and I am going to replace this reduction while a production which is the markers and then all can go M_n in the stack and then this is attributes goes off X_1 s is going to empty values that corresponding to X_1 and derived that X_1 is going to the marker corresponding to S okay.

(Refer Slide Time: 21:20)



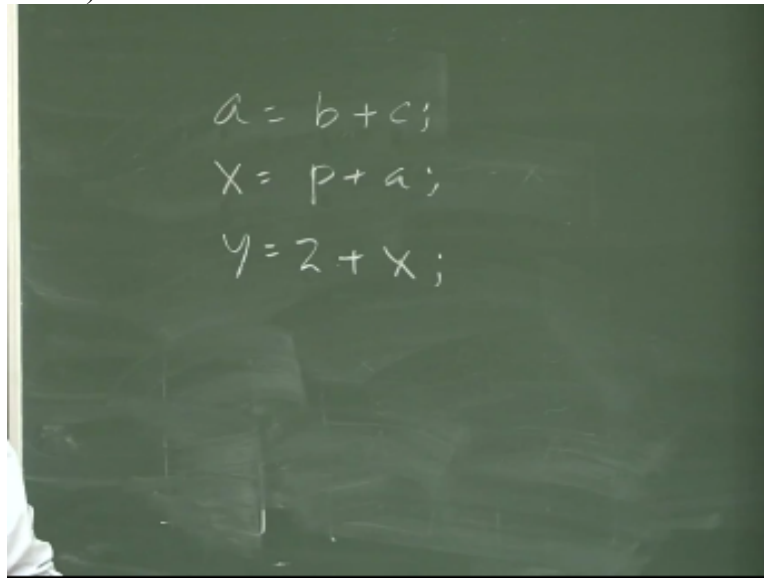
That become using the notation so if reduction to a marker then marker belongs to this particular production and what will I do A_i is in this position this top minus $2j$ plus 2 okay this is the computing form with this particular stack and $X_{1,i}$ is in position which you can solve which avoid and so on okay and the reduction is to a non terminal A by this production then what going to compute A_s and push on the stack okay.

So this is the most gentle schema have for finding out or are attributed and the definitions on the stack itself of course I am assuming here that every grammar symbols will have only one in there because basically what we are doing is we actually in space understand so this can be easily generalized yeah any questions on this okay so let us continue our discussion on the optimization okay I think to worry about keeping all the attributes some we right certain space was hesitate one once space I have created all these are markers.

And I am saying that sound good enough to say that and therefore use markers only for the symbols and therefore use markers only for the symbols which requirement inherited.

A few and put them on the right you can do better than so now we are really worried about the implementation we really want to have a compiler which is fast one would have the fast can I further use say something make a solution okay so how do I do that so proposal is that all markers which hold the same value.

(Refer Slide Time:24:25)



So let me give you a program instance good can I go for the better in that so let me give you a programmers now suppose I have a program that is nothing X how I variables I have so I have a one two three four five six and seven right so where are do an memory of an location is I am going to sat that I am locate seven locations are there why any need seven locations so what happens here is and I am saying that a is b assign b plus c now.

the sound location is associated with b and c but you can see that beyond this point this b and c so can I just say that let me over right some other locations and use that for A can I do that okay similarly when I say know that p is in location when I say that I and use for the a location for b and then p location for c that stack is not require and then I say that beyond this point by p and A are require so X takes this two locations.

and then the locations is used by Z and then for already location that assign to y so we need seven or I can go with queues and use this two and three so what is that I am using that so let us I am doing this then I am looking at life time of vertically I am looking that life time of variables starts with symbol with initialized this time to last used and beyond this I want to have a grammar space on this I can free this and same thing I am going to use this no reason that the attribute at the copy of each other and the attributes beyond the point.

The attributes depends are something to do similar to do this right so if I define this attributes beyond the point from the computation right way to keep it on the stack I t need blocking all the information on the stack okay I can as well as registers I can use some other locations which are not even related to the stack and say that my attributes are going today okay if I say that life time reputations.

(Refer Slide Time: 26:52)

Space for attributes at compile time

- Lifetime of an attribute begins when it is first computed
- Lifetime of an attribute ends when all the attributes depending on it, have been computed
- Space can be conserved by assigning space for an attribute only during its lifetime

so life time begins where it is first computed and life time attributes ends and all the attributes depend up on it I will become to this okay and as soon as I have sign that is release the stack okay for example in this case life time of B and b and c and as soon as this state meant is computed this beyond this point b and C use so I can use the same location so state contents the are the having space attributes only do it is assigning space four of the rules okay.

(Refer Slide Time: 27:21)

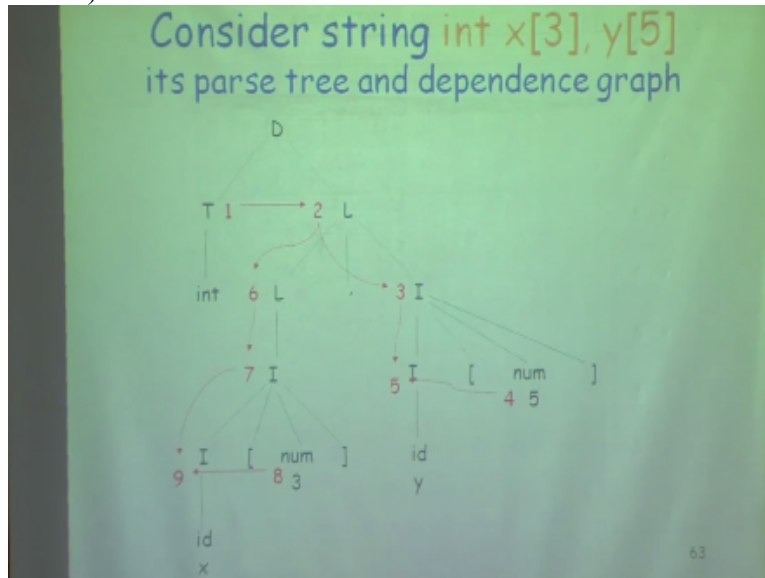
Example

- Consider following definition

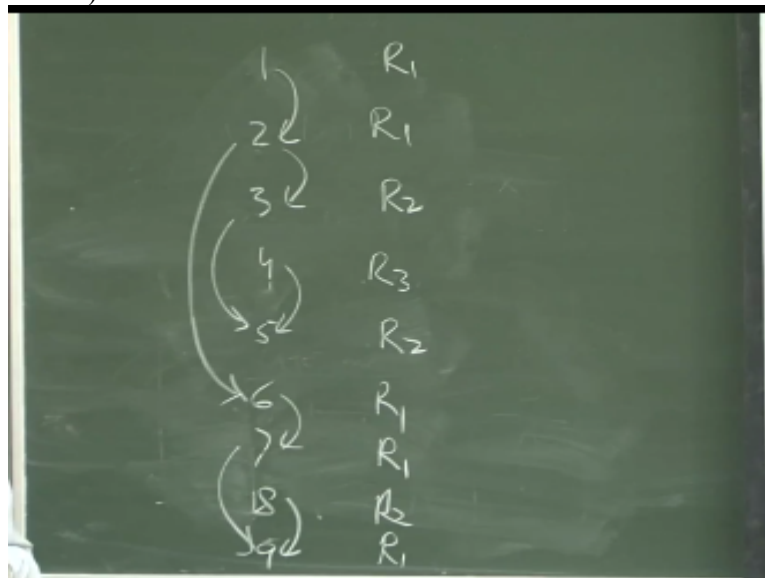
$D \rightarrow TL$	$L.in := T.type$
$T \rightarrow real$	$T.type := real$
$T \rightarrow int$	$T.type := int$
$L \rightarrow L_1, I$	$L_1.in := L.in; I.in = L.in$
$L \rightarrow I$	$I.in = L.in$
$I \rightarrow I_1[num]$	$I_1.in = array(numeral, I.in)$
$I \rightarrow id$	$addtype(id.entry, I.in)$

So suggest to this computation okay let us look at the grammar which is likely similar to what we have done do not worry about the grammar okay that is not important part okay but I am doing issues attributes computation of the grammar and I will find the different are the last okay so

what was the rule is here I am initializing or my identify of have to right solution but I have also introduce now that I am saying that I can have it will more complex okay.
 (Refer Slide Time: 27:50)



nothing more detail but important part is that if I now trying to past in X3 X5 and 5 and I will look at first abstract syntax tree for this and then I introduce all the dependencies this is dependencies going to be directly what is going to we use for computation of 2 and 3 and 6 are going to use for what are the computed is here and 7 is going to be use 6 is on okay so this all computation are going on okay and now let us see what is the lifetime of these variables and how do I optimize my number of locations.
 (Refer Slide Time: 28:51)



So let me just put this computations and I am going to computations form 1 to 9 so I have this computation 1, 2 3 let me right vertically so that more space so these are the 9 computations and

then okay so I am looking at the define this all of here he say that 2 is one of computing one and then 6 and 3 use purpose computing 2 and 7 users computing in 6 and then 9 use as computed in 7,8 and the other side the 5 users what was computed in 3 and 4 right we just setting all these right all those and flat in this on so now we can see that as for as this is concern so let me start allocating resource to this so I can say that first item compute one.

Therefore I will allocate resource on this and then a compute but now we are beyond this point one and two does and therefore I can use the same resources okay R1 at the I say that I want to compute 3 it is depend up on two now for okay but I can see that for before four can be given a space I cannot see this space and I cannot feel this space they have a future use why lose compute to use and to compute to use say that I'm going to use an r3 here and now when I come to this node and say I want to computation then I find that beyond this point a both r2 and r3 have no use and therefore I can read both these locations and I can say that this goes into resource R 2.

And then I compose this node and now I find that this node depends only on this and now I can see even and therefore I can allocate to the R1 source number six okay and when I come to seventh then I find that seven depends on six which is bound to resource r1 so I can and then I can compute for this information R1 and then find that our move beyond this point I had no use finally computation was done.

so when it comes to our gate I can use r2 here and when it comes night I can feed both r1 and r2 and put that information R1 in so rather than having a large stack when I was putting all these grammar symbols I can actually do this right I can do better than this remember that we are trying to extract every possible information look for everything this all the information is in this stack so let us go back in this.

so what happens here when you say L heading this P type II type is video 3 x is inferior and one inherent I inherited L inherited so I okay if you see all these are nothing but pops okay only computation that is happening is either here or here rest time just now if I just have creating copy now you can see that actually who is nothing but if I say this is my resource r1 who is nothing but R1 the same pop okay let us what I did if we go back then it will see this type i just need to the pop okay.

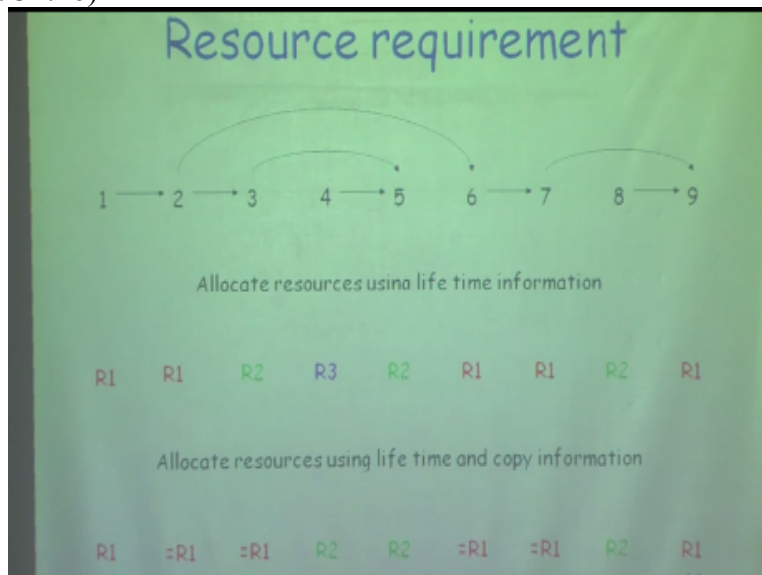
If I am creating the another copy then I go to no worry about work like that so find out what are the copy which can find out so I created copies here then 3 & 6 also happened to be copies of R1 3 & 6 name of the non terminal form okay and what about 7 also happens to be a copy of r1 and

so if I use this information okay only computing that is happening is that 5 in line and all are the resources will be just one right.

and I first compute by put that information with the similar table and corresponding to 9 I will do this computation that I can actually for this two resources so I want to maintain that large stack by using this other locations right using all possibilities life time informations finding out what are the rules therefore backing the resources we did so and remember that all the strings are going to be used for the rotation when I finally start the machine code.

We will find that I have giving a number of we will decide that is most machines is providing way processing this way this registers and I will trying to see that whenever I can do that an expression and trying to do this registers and I try to see registers as well as possible whenever there is no creature loose and if it is keeping the same values the same picture here and say that if I look at resource requirement this is how my graph looks okay.

(Refer Slide Time: 34:16)



And if I allocate resources using lifetime information this is how the information is going to be bound and finally I am using only 3 locations and I am using allocate I am trying to not using information by copy information and suddenly I find that many of these are copies and if they are copies then I just need to resources so whatever information is available to you try to use that information to further optimize so all you can see that not only we understand and welcome we understand where the attributes are available.

But we can also further optimize it and see that I can now work with just two resources which happens to be in this case but the techniques are general you say that whatever our copies might certain variables do it goes and then we use lifetime information and try to minimize all this of

the number of resources okay so some rules are that you will just look at copies and we look at lifetime and once we apply this control then we will be able to further optimize.

And of course number of resources is depend upon the kind of program that I am different so for different grammar rules we will have different number of resources, so copy information is right there like you see that if you go back to this grammar here so let me go back to this grammar here all I can go back to this form done okay this is suppose I found all resource will be back this now you can see that when I am talking of a detective I am making the copy so I can say that look lifetime of T type is not the standing here because I am here .

Similarly if I look at hesitate is being copied there so I can say this is basically the same resource I do not have to be again so by finding out information and I kept sending my lifetime information so this is what happens if you look at last we are saying that who is a copy of one and six and three are also copies of two and seven is a copy of six and five happens to be a copy of five we depend upon this but so you can see one two three and four.

These are nothing but copies and this is where I use this information that I say that one is and then so one, two, and three and seven are not but copies one so that means I can say that whatever was allocated here is being used here and therefore gone three R1 one in this one once I recognize this fact then you will find that what I was doing here I was saying that I am going to use this resource and then use R2 but this is nothing but a copy right.

So I do not have to allocate the new resource for this is already there and then I say that as far as this node is concerned compute to that in our two but life time of R2 hands here because five is depending upon three and four and therefore it not really a copy it requires so whenever you have a file and not pop you need a new resource whenever it is just copy of your another then I can lose this copy just okay .

So that means finally depend upon three and four and three have the up for their rule which is R 1 but 4 can we free here and therefore fight will going on so similarly now once this part is done then I say it is not a new resource required but 9 now both R 1 and R 2 can be free because beyond this point given R1 there is no use the compiler need to worry about space okay so you are computing the compiler you need to worry about space so I have told you this story somewhere in the beginning where people .

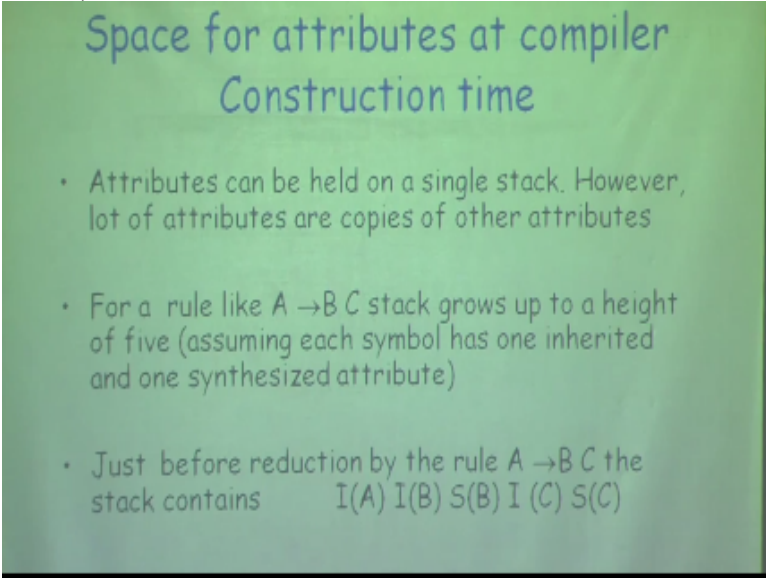
So we had to make a compiler in certain manner we found suddenly that the compilations becomes so slow the complication of just coming out it is not to the compilation out the reason are the simplification became so large and all the time you have sufficient memory can doing this stack I will put some space in memory which is kind of minutes were we said and that and the virtual space is having to the usability.

And therefore we have internal memory of 4gb or 32 gb now we are talking about memory kind of and then certainly you found you have similar table which is not put in and all the space using this programming you want to compile the function which are going to the part okay so people will not carefully build this single table format and once this go to the optimization then formed that because the matter of the sets okay.

Because remember that the I comes over access not just in term of that was symbol table can be manipulate are and the linier single tables are started using and so on we have to use this side it does not fitting to the way then I going to operating system is going to waste of time to be aware of this stack that was making compare the systems software is should on a control of operating system and few and if you are not careful space optimization.

So this many of the people have this motion saying oh , do not worry about space non of never non of memory on this and always right program which can clauses so one has to be very, very careful and one is to learn all possible techniques will reduce among the memory, memory is and the sources are all the three , so this is one of imagination okay but to further optimize.

(Refer Slide Time: 40:44)

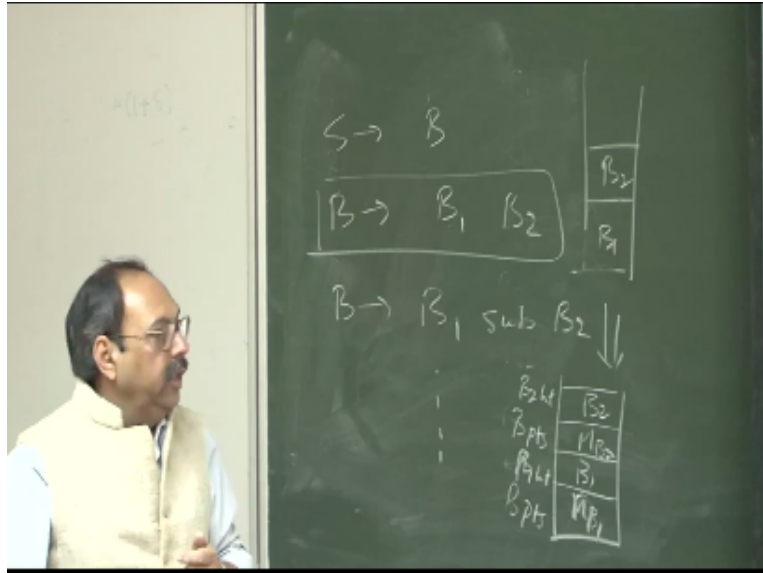


Space for attributes at compiler
Construction time

- Attributes can be held on a single stack. However, lot of attributes are copies of other attributes
- For a rule like $A \rightarrow B C$ stack grows up to a height of five (assuming each symbol has one inherited and one synthesized attribute)
- Just before reduction by the rule $A \rightarrow B C$ the stack contains $I(A) I(B) S(B) I(C) S(C)$

So let me go back to one of the examples he dealt with earlier.

(Refer Slide Time: 40:55)



So we had this example which said $S \rightarrow B$ and then we had this block going to B_1 or B_2 and then we had another block going to B_1 subscript B_2 and so on, you remember the example what was this right when I start looking at one information here what happens is that attribute can we held on a single stand so what, what where what is that we were doing that when this induction was taking place basically what was happening was that I had this D_1 on the stack then I had D_2 on the stack now suppose I have to put market on the stack then I also need space for inherited attributes.

So when I say space inherited attribute that means immediately my stack change to something like this I must have had a marker for B_1 then I have B_1 then I have a marker for B_2 then I had B_2 my actually stack was siso right and these markers was keeping the heldier attitudes what was the heldier attitude in the example point size and what was the synthesize attitude size right so I was keeping now point size there and I was eating of even height there.

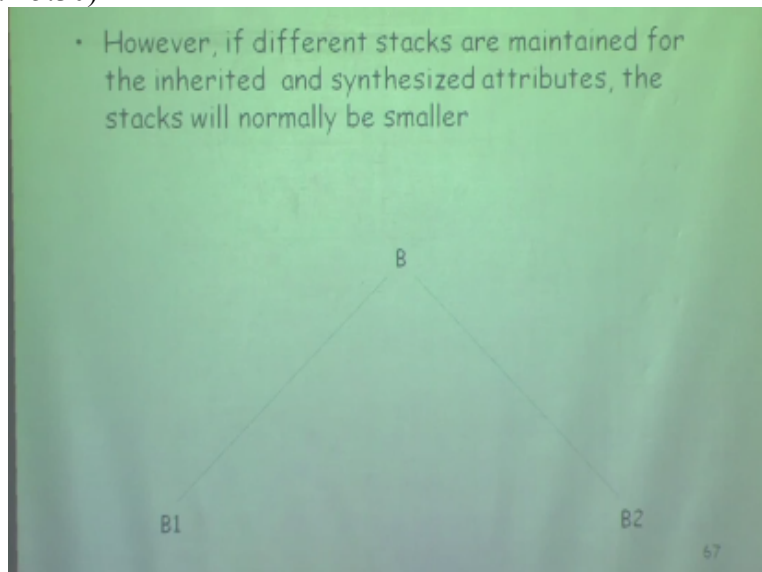
Now you can see that actually these are nothing but hobbies so one way I just showed you that I am going to use the discus and with you certain resources I do not want to use stack but sometimes you will not be able to avoid stack but can I split this now, now what is the so when I say that I am going to do a reduction by this rule of this form what is the maximum size of the stack possible potentially what I can do .

So when I say I am doing reduction by this rule let us say forget about the any other symbol okay what can be the maximum height of the stack here four right so in general if you have the symbols in the right hand side you can height of the stack able to be height right but you find within this two K you have these hobbies yeah, suppose I separate my heretical statistics in the

stack itself then what will happen I will have two specks of fight on but as soon as I do this separation.

I evaluate B - I will also have B to fight on the stack so it goes and then and sizes and only two will have both synthesized and then finally this reduction happens and now you can see that my inherited at you because of all these coffees and my synthesized that you starting from empty it to and so you can see that if I separate this acts.

(Refer Slide Time: 45:30)



One for in acted another goes inside if I do that then what will happen is that I will say that B point either here but B at synthesize here and now I say now this is going to becoming. Using the copy whose I say who I know for show that B1 point are nothing but B points at that. So do not have pushed up that is already in the form of N right. And then I am going to save at I only read to compute B1 height here E1 points already in the form of fact.

And again I find B2 points is nothing but a copy you will top of style and only my since has go to which will have both synthesize we will put on B1 and synthesize of B2 and then finally this should be happens and my synthesize become right one. Now you can see that my in hesitate attitude is that did not growth the on one. Between only the 51 because3 of this copies okay. And my synthesize you started with empty in group to 51 in used.

now I mean this in percentage-wise this may look I mean I may not look very large percentage will do this computation but really difficult um when you find a lot of production that So crates the synthesize meant an editor you have already out together and here I have found the stack which is a white one I could have said that this one here all the time so that we want it will come back separate instance right so I just want to see the principal's there remember two things one

that is pretty to me certain attributes we have this copy of each other and therefore you should preserve that information what is coffee will be dinner and second that each at the right time and allocate resources only for the duration of lifetime and not others if you just remember these schools the space requirement is going to go down drastically.

And these are the tricks which are used in this and so I mean I do not want to distinguish between the discussed and resources so these are nothing but as far as we are concerned these are resources and to optimize is the one you find you have some addition to be done because now you want to manipulate all the stack conditions that you must remember but once you have done that x1 time exercise when you find that compilers even faster but to worry about compile time it is only one time activity and this has an effect on my screen time house because if you use too many resources suppose.

I do not optimize my poll I just use large number of registers what will have them I have to do this stress varies of it very successfully means extra machine cycles and if there are more instructions that means except X items so a good compiler is going to make sure that you get a copy my scope but with the same fact remember compiler also program so compiler itself needs to be optimized any questions okay so let's break here today and tomorrow we are going to start our discussion on fight systems and see that how to compute all the time expressions and here and continue efficient tomorrow.

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

@Copyright reserved