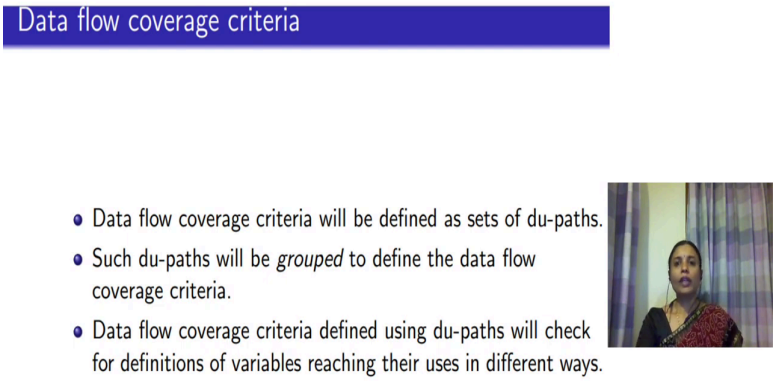
Software Testing
Professor Meenakshi D'Souza
Department of Computer Science Engineering
Indian Institute of Technology, Bangalore
Algorithms: Data Flow Graph Coverage Criteria

Welcome back, we continue with data flow coverage criteria in week 3. Now, last time we defined what data flow was, and what is the definition of a particular data, what is the use of a particular data, what is the du-path. Now, based on these definitions, we are actually going to see coverage criteria that work with these data flow definitions.

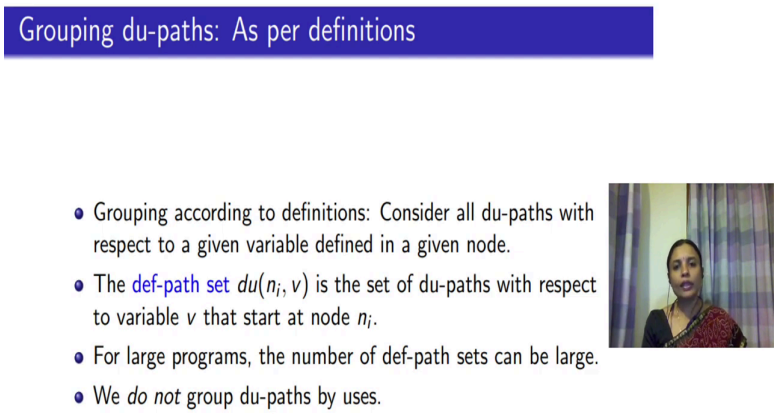
(Refer Slide Time: 0:43)



So, we define data flow coverage criteria as a set of du-paths, paths that take a variable value from its definition through its use, as the program executes using a sequence of statements. And sometimes for large programs, typically, they tend to be large, there could be many variables, they could be defined and redefined at several different places. And there could be several different du-paths.

So, while looking at criteria, it helps to group the criteria using some way or the other. So, we are going to take the du-paths that take variables from definitions to use, and group the du-paths in some way or the other to define data flow coverage criteria. How are we going to group them?

(Refer Slide Time: 1:38)

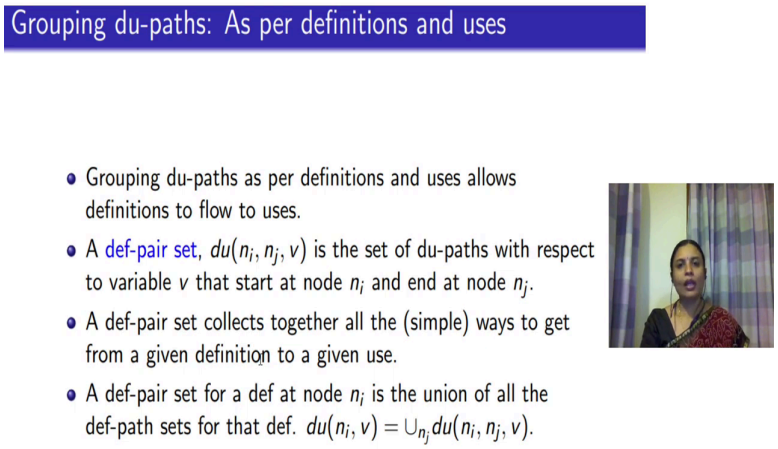


We are going to group them like this. So, we first take a variable, let us say one particular variable, fix it, let us say variable v and consider all the du-paths for that variable from a particular given node. So, that is the variable is v, the node is ni and I am going to consider all the du-paths that begin at that node for the variable v, we call a def path set, short form for definition path set, write it as du(ni ,v) as the set of all du-paths with respect to the variable v that start at this node ni.

We had already fixed the variable v, we are considering all the du-paths for the variable v that begins at the node ni. And as I told you, for large programs, such number of parts can be large. So we group them once again, we might want to group them by uses but it is typically not done in software testing and when we look at examples, it will become clear to you why we do not group them by uses.

So take a def path set like this, that is for a variable v, all the du-paths that begin at a particular node ni.

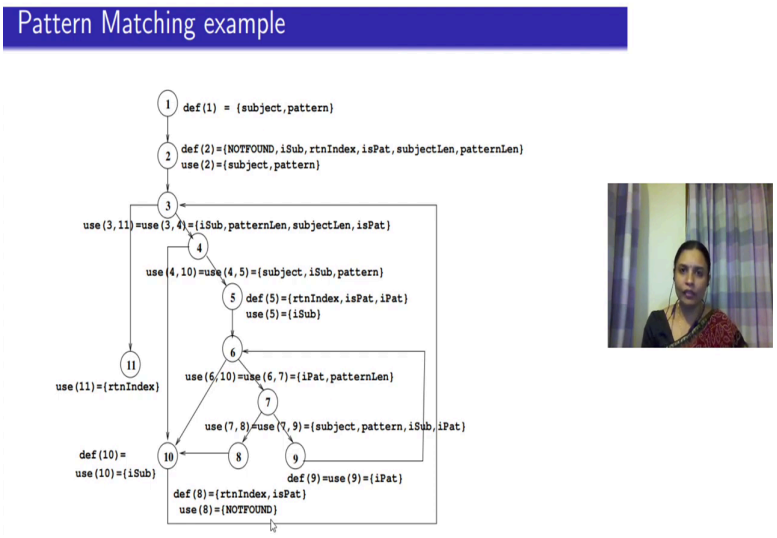
(Refer Slide Time: 2:57)



So, we group du-paths by definition and by use and this will allow the definitions of the du-paths to flow over to its use as the program executes. Such a grouping is called a def pair set, short form for definition pair set. The definition first set du, the same v is there, same ni is there, now we fix a end node nj, so du ni nj v is the set of all du-paths with reference to the variable v that starts at node ni and ends at node nj, that is the def pair set.

So a def pair set basically collects together all the ways to get from the definition of a particular variable v to a given use the same variable at a particular node ni. A def pair set for a definition at node ni is nothing but the union of all the def path sets for that definition. As they come in different-different places of different-different uses that vary over nj, that is considered variable v that is defined at ni and consider all possible uses that end at nj. So that is du(ni, nj, v) and take its union over all such nj, then we get this entire def path set.

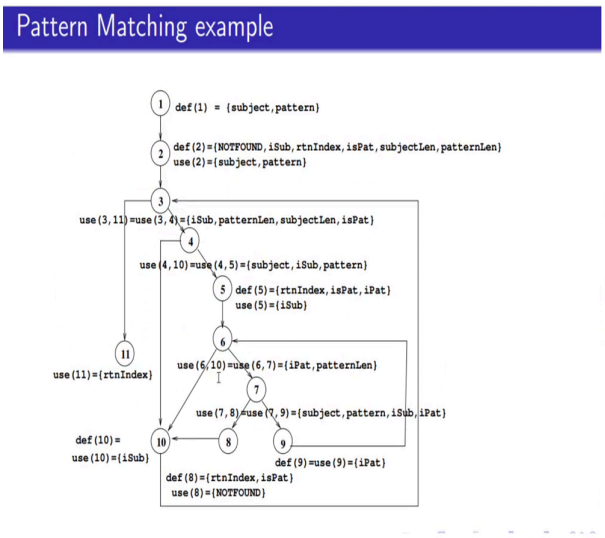
(Refer Slide Time: 4:32)



We will go back and look at the same pattern-matching example. I am not showing you the code once again. It was there in the last lecture. We will directly consider the data flow graph of the pattern-matching example. The data flow graph is given to you here in this slide. What is happening here? If you see, the same CFG with augmented with definitions and uses.

I am going to illustrate to you what these 2 definitions are now on this example. What is the def pair set? And what is a def path set?

(Refer Slide Time: 5:03)

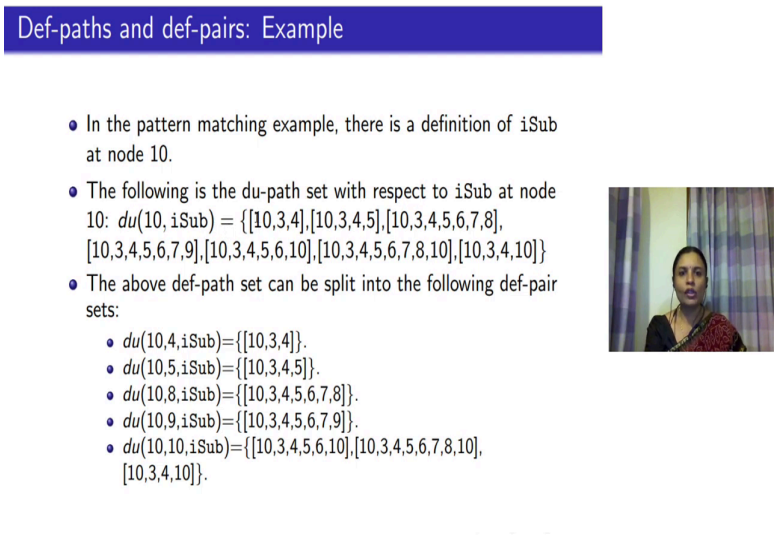


So for illustrating that, let us take the same pattern matching example and look at one definition of let us say, iSub at node 10. So where is node 10 in this figure? It is here. And a

variable iSub index over subject is defined here. It is also used here, but it is used in several other places. Can you spot one more use of the same variable somewhere else down the control flow graph?

So let us say 10, there is a path back to 3. And there is a use here in the edge 3 4 of this variable iSub, and then there is another use in the edge 4 5 of the same variable iSub, there is one more use in the edge 5 of iSub. And there is one more use here, down below in the edge 7 to 9. You are able to understand? So go from 10 to take us back edge, go from 10 to 3, 3 to 4 edge, there is a use, 4 to 5 there is a use again, at the node 5 the research was and then go 5 6 7, the edge 7 to 8, there is one more use of this variable iSub that is defined at 10. So that is what is basically listed in this slide.

(Refer Slide Time: 6:26)



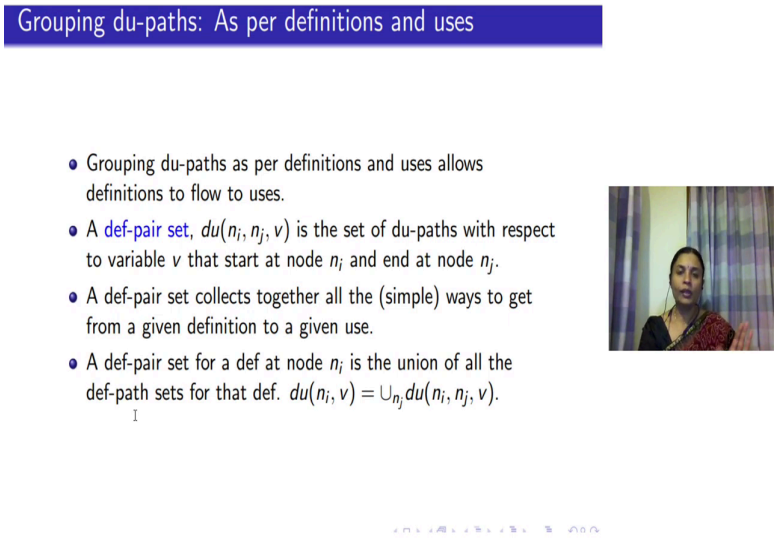
In this example, we just saw that there was a definition of iSub at node 10. In fact, collect the du-path set for iSub at node 10, then this would, this whole thing would be the set. This 1 that goes from 10 to the edge 3, 4, as we saw; 10 to the edge 3, 4, to the definition at node 5, and then down all the way to 6 7 8 6 7 9 and then back to 10 in 3 different ways.

So this is the whole thing, which is the du-path set. I can break this du-path set by grouping together. So here what did I consider? I considered the variable v was iSub and the node that I began in was 10. So this is du ni v and uses at various ends were listed here. Now I can break down the uses also. So I can say, do du 10 4 isub, this is for the variable iSub, the definition is at 10 the use is at node for.

That is this du-path 10 to 3 to 4. For the same variable iSub, definition at node 10 ends in the use at 5 in the du-path and to 3 to 4 to 5. For the variable iSub the definition at node 10 ends in the use at node 8 through this path, du-path 10 3 4 5 6 7 8 and so on. I have just basically taken this set above here, the set of du-paths for iSub that begin at node 10 and broken it down into the set of du-paths for iSub that begin at node 10 and end at all these various nodes.

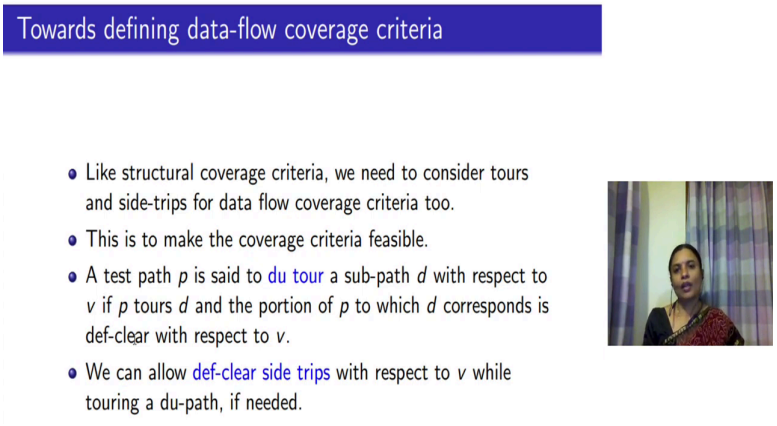
So, if I take the union of all these parts here, I get the set above here, that is what is given here in this definition. def pair set and def part set. So is it clear?

(Refer Slide Time: 8:34)



Basically there is nothing, no magic here, we will take a variable, take a particular definition of a variable at a particular node. If you consider all possible du-paths, then you consider all possible uses ending at all possible nodes or you could say for this variable with the definition beginning at this node, I consider use at a particular node nj and that becomes a def pair set. So, this is another kind of grouping.

(Refer Slide Time: 9:03)

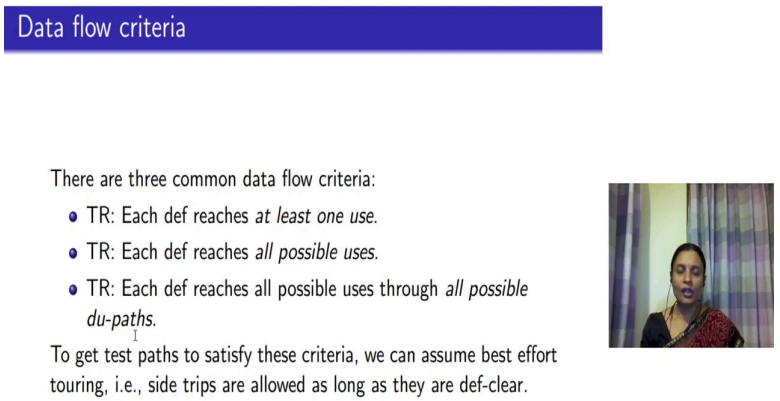


So, let me go back down below. Now we are ready to define data flow coverage criteria. So sometimes you remember when we did structural coverage criteria to make prime path coverage especially feasible, we had to sometimes look at side-trips and du tours. Here also we can look at side-trips and du tours. In graph coverage always side-trips and du tours are permitted to make coverage criteria look feasible.

So a test path p is set to du tour a sub path d with reference to a variable v if the path p towards d in the portion of p to which d corresponds to, is def-clear away as far as variable v is concerned. It is the same thing. It is the same 2 array. We just ensure that as far as v is concerned, it does not get redefined along the du tour. Same thing for side-trips also, we ensure that it stays definition clear as far as the side-trip is concerned.

Otherwise, we allow both side-trips and du tours as far as they do not redefine the variable v that we are focusing on.

(Refer Slide Time: 10:16)

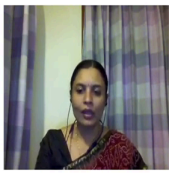


What are the 3 data flow criteria? They are very simple. Data flow is all for a variable, what its definition is? What its use is? And the flow of the value of the variable from the definition to the use. The data flow criteria also talk about the same thing. The 3 common data flow criteria are, first one, test requirement, each definition reaches at least one use. It does not go waste. Every variable that is defined is used somewhere at least.

So, write a test case that ensures that every definition reaches at least one use of that variable. So, that is the simplest data flow test requirements. The next one is more ambitious, it says every definition of every variable write test cases that reach all possible uses. So, your test requirement is each definition reaches all possible uses. The third TR says each definition reaches not only all possible uses, but it reaches all possible uses through all possible du-paths. Meaning, so, you could reach the same use through different-different parts.

(Refer Slide Time: 11:36)

- In the pattern matching example, there is a definition of iSub at node 10.
- The following is the du-path set with respect to iSub at node 10: $du(10, \text{iSub}) = \{[10,3,4],[10,3,4,5],[10,3,4,5,6,7,8],[10,3,4,5,6,7,9],[10,3,4,5,6,10],[10,3,4,5,6,7,8,10],[10,3,4,10]\}$
- The above def-path set can be split into the following def-pair sets:
  - $du(10,4,\text{iSub})=\{[10,3,4]\}$.
  - $du(10,5,\text{iSub})=\{[10,3,4,5]\}$.
  - $du(10,8,\text{iSub})=\{[10,3,4,5,6,7,8]\}$.
  - $du(10,9,\text{iSub})=\{[10,3,4,5,6,7,9]\}$.
  - $du(10,10,\text{iSub})=\{[10,3,4,5,6,10],[10,3,4,5,6,7,8,10],[10,3,4,10]\}$.

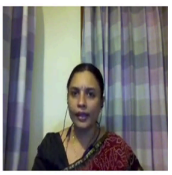


Data flow criteria

There are three common data flow criteria:

- TR: Each def reaches *at least one use*.
- TR: Each def reaches *all possible uses*.
- TR: Each def reaches all possible uses through *all possible du-paths*.

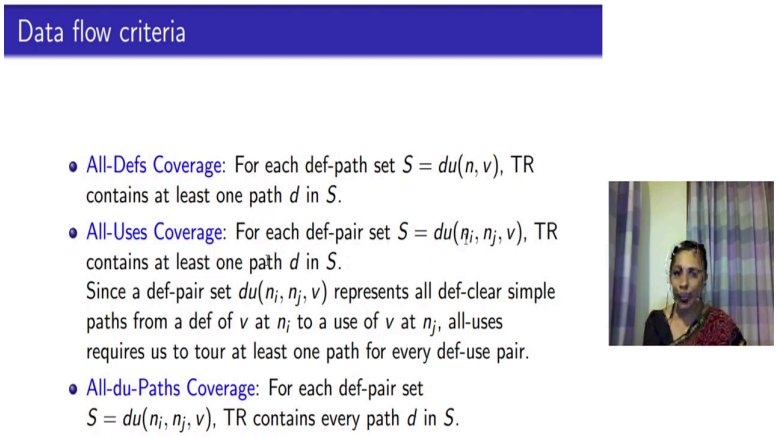
To get test paths to satisfy these criteria, we can assume best effort touring, i.e., side trips are allowed as long as they are def-clear.



Let me just go back and show you this example. If you take this last one, the definition at 10 reaches the use at 10 for the variable iSub through 3 different paths. Is that clear? So, what the test requirements says is take all the 3 paths, that is all the third test requirements. So, to summarize, there are 3 data flow coverage criteria, first one test requirements there is every definition reaches at least one use.

Second one says every definition reaches all possible uses. Third one says every definition reaches all possible uses through all possible du-paths. And as I told you just in the previous slide, if you encounter any of these test requirements to become infeasible, allow side-trips and du tour as long as they are definition clear.

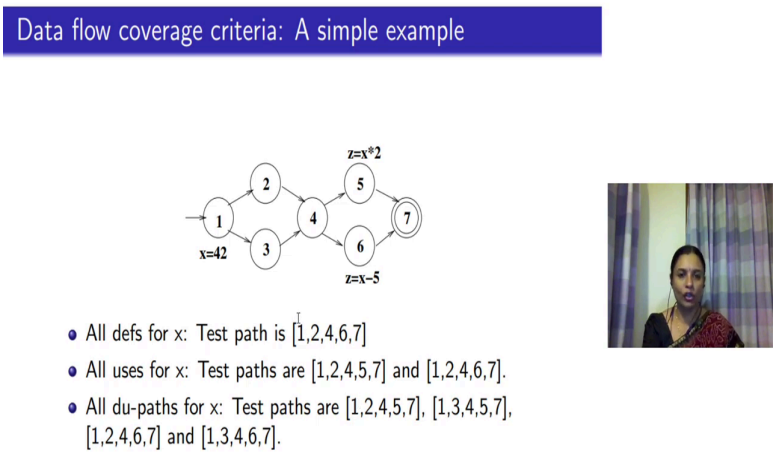
(Refer Slide Time: 12:32)



The same data flow criteria that I have defined formally in this slide all definitions coverage, test requirements says for each def path set du of the variable v at the statement n, TR contains at least one path d in S. All coverage says for each def pair set, du ni nj v, TR contains at least one path d in the set S, that is at least one definition in ni to at least one use in nj.

All du-paths coverage says for each def pair set, du ni nj v, TR contain every part d in the set S. So, the first test requirement says every definition reaches at least one use, second test requirement all uses says every definition reaches all the uses. Third requirement all du-path says every definition reaches all the uses through all possible paths. I hope the 3 data flow coverage criteria definitions are clear to all.

(Refer Slide Time: 13:46)

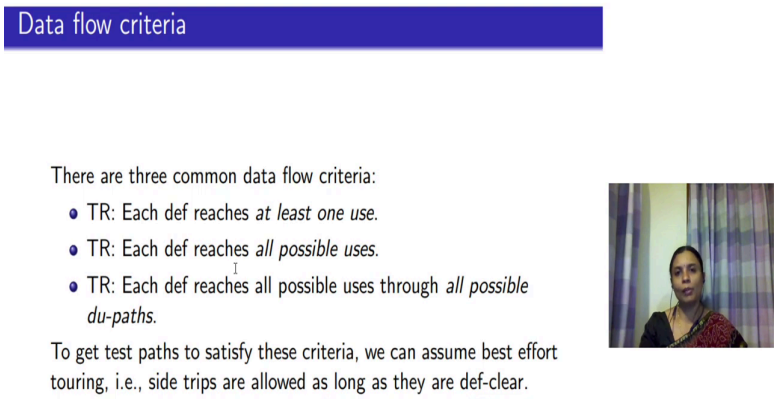


So if you remember we had the simple example in the last lecture, when we introduced data flow coverage criteria, where x was defined at the node 1, z was defined at nodes 5 and 6 and x was used at nodes 5 and 6. So if I have to achieve coverage criteria just for this small example, if I have to say all definitions for x, that is every definition reaches at least one use, so the test path would be begin at node 1, x is defined here and maybe go through node 5 or node 6.

So in this example we have gone through nodes 6 1 to 2 to 4 to 6 to 7, we will ensure that the definition of x at 1 reaches at least one use which is at 6. All uses says the definition of x at 1 should reach all the possible uses for x. Which are the 2 uses for x? one at node 5, one at node 6. So all uses the test requirements for x, the test paths are 1 2 3 4, they use at 5 ending at 7 and 1 2 3 4, they use it 6 ending at 7. Is it clear?

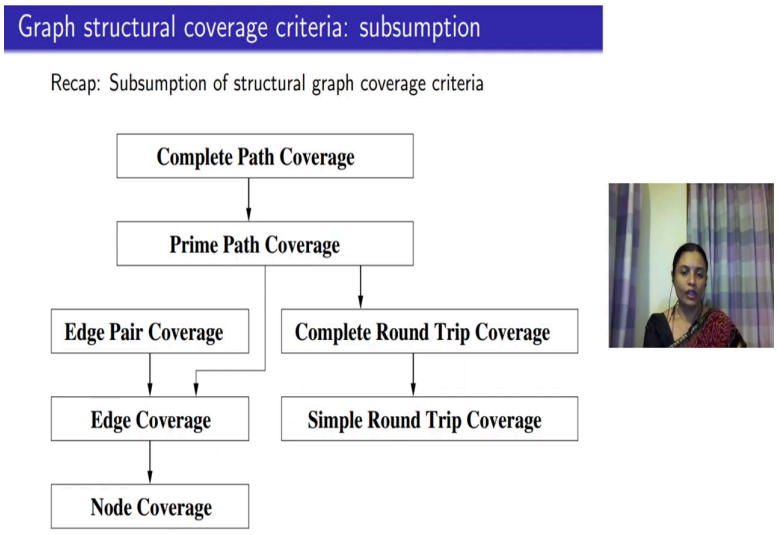
Alternatively, you could consider 1 3 4 6 7 also. The third one, in fact does that. Third one is the strongest data flow coverage criteria. All du-paths for x, that is, you go from definition of x to all possible uses of x through all possible paths. So to reiterate definition for x is at node 1, and the uses are at node 5 and 6, and all du-paths test requirements basically lists the 4 possible paths that take you from the definition of x to the uses at 5 and 6. So the 4 paths are 1 2 4 5 7, 1 3 4 5 7. Similarly, 1 2 4 6, and 7 and 1 3 4 6 and 7, is it clear?

(Refer Slide Time: 16:08)



So let me just summarize it quickly. This slide is a useful summary, 3 kinds of data flow coverage criteria requirements. Every definition reaches at least one use. Every definition reaches all possible uses. But in the second case, I do not worry about which is the path that I take. In the third case, I worry about that. I say every definition reaches all possible uses through all possible, that is what is the third test requirement. Is this clear?

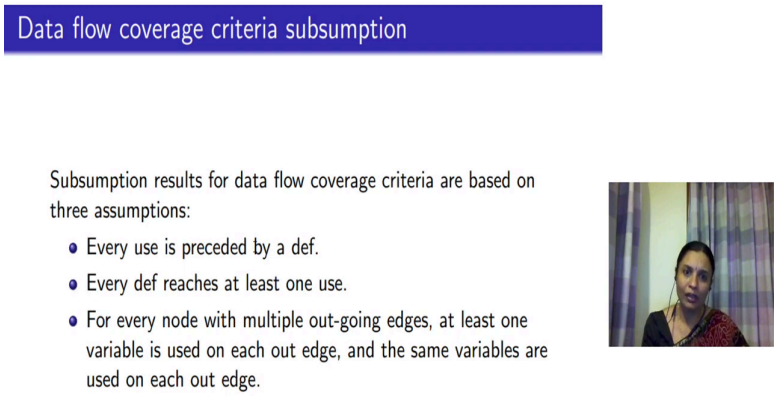
(Refer Slide Time: 16:38)



So let us go down. Now is the time to look at subsumption. If you remember, when we had looked at structural draft coverage criteria last week, these were all the coverage criteria that we looked at, purely based on the CFG node, edge coverage, edge pair coverage, prime path coverage, then complete path coverage, and then we had round trips, simple round trip and

complete round trip. So this is just a recap of what we saw earlier. Now we saw 3 new data flow coverage criteria.

(Refer Slide Time: 17:10)

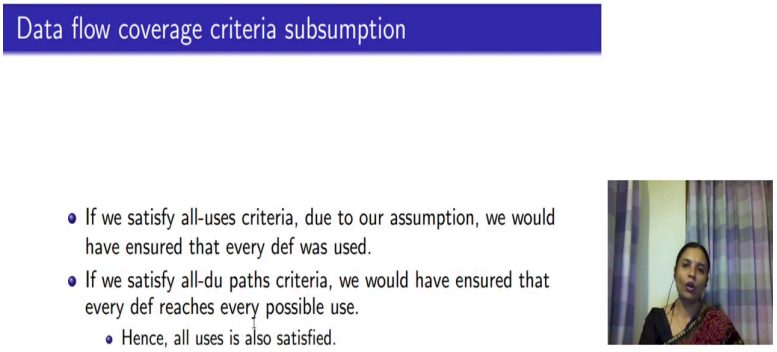


So we will first see how they subsume each other. Before we see how they subsume each other, a small set of assumptions are important to make sure that we do not get anything unsolved. So the subsumption results for data flow coverage criteria are based on 3 assumptions. One is, we assume that every use of a variable is preceded by a definition of the variable somewhere.

Obviously, if a variable is going to be used, it has to be first be defined, otherwise, even a compiler will catch the problem. So we assume that. We also assume that every variable that is defined reaches at least one use, it is not left unused at all, we assume that this property holds. And the next assumption is if I have a node in a control flow graph with multiple outgoing edges, then at least one variable is used in each of the outgoing edge and the same variables are used on each out edge.

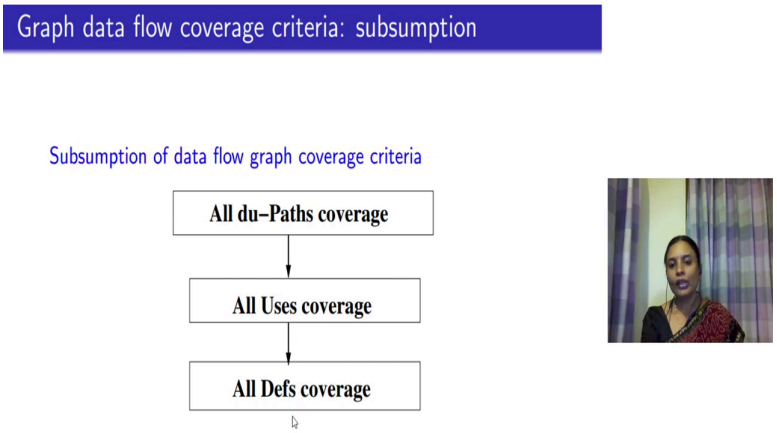
I should be able to get the set of variables that are used in the outgoing edges. These are simple assumptions that I need before I look at data flow coverage criteria subsumption.

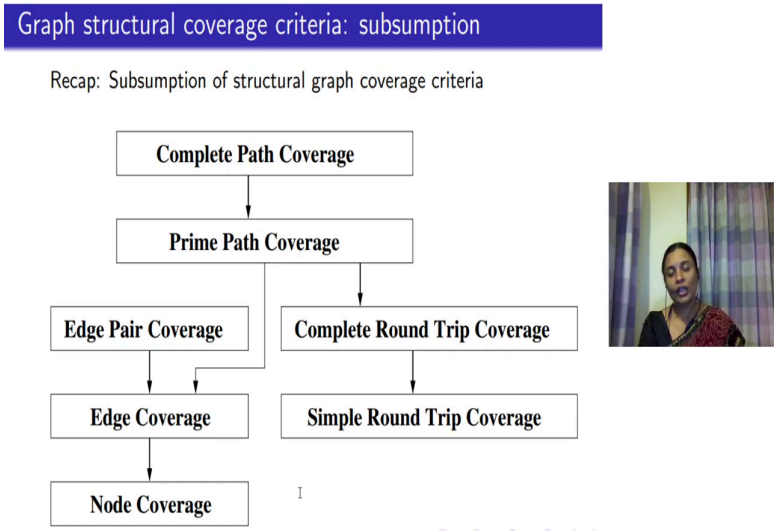
(Refer Slide Time: 18:28)



So, the subsumption is fairly easy to see, because the simplest will be what? Every definition reaches at least one use. So which means if I satisfy all uses criteria, that is every definition reaches all possible uses, then I know that every definition reaches at least one use. So it obviously subsumes all defs criteria. Similarly, if I satisfy all the du-paths criteria, that is every definition reaches every possible use through every possible path, then I am obviously including all uses criteria. So, this picture should be fairly clear.

(Refer Slide Time: 19:10)

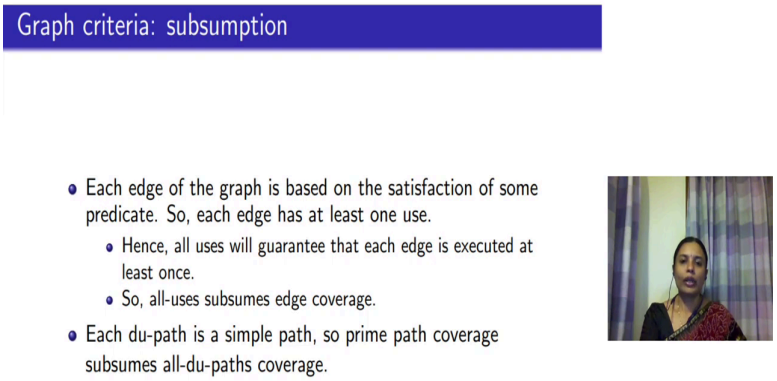




All du-paths coverage is the most expressive data flow criteria. It just basically says for a variable, take its definition, take all possible uses and take all possible paths that go through various uses. This is going to subsume all uses coverage, which in turn is going to subsume every definition reached at least once, that is all defs coverage. So this is one picture of data flow coverage criteria subsumption and this is the other diagram of structural coverage criteria subsumption.

They both work on consume control flow graph, one considers the data flow information, one ignores the data flow information. Now can we compare these 2 coverage criteria and say how structural coverage criteria compares to data flow coverage criteria, that is what we would like to see.

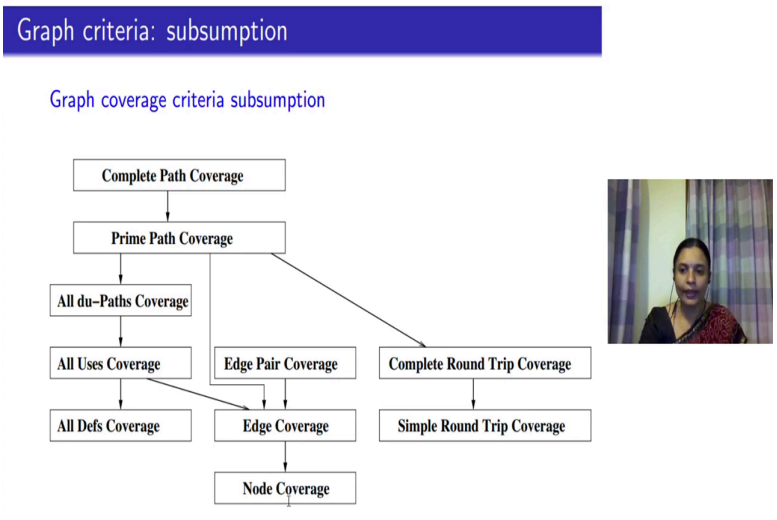
(Refer Slide Time: 20:08)

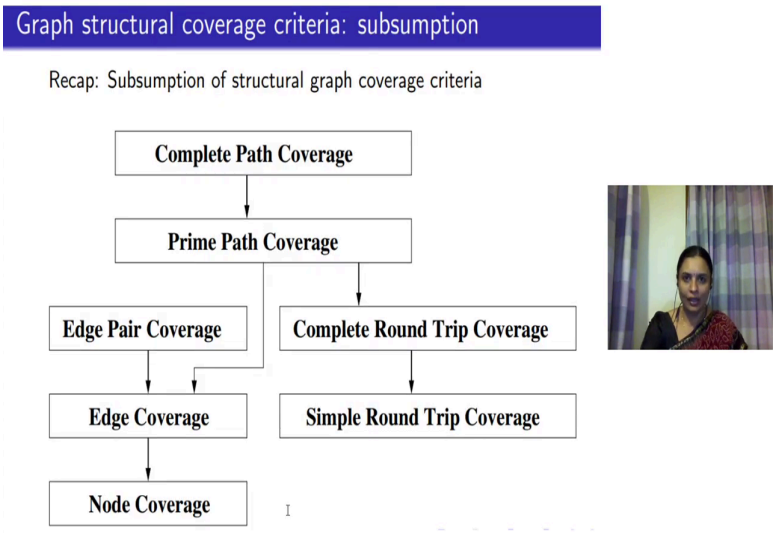


So, each node, each edge of a graph is based on satisfying some predicate, especially if I have if and while statement, the relational logical expression that is going to be true or false. So each edge will have at least one use, there is going to be a predicate, which uses the variable, makes it positive or negative, tests for it and all that stuff. So definitely, if I do all uses data flow coverage criteria, I will do edge coverage criteria for sure.

Similarly, because of my assumption, that each du-path is a simple path, remember, we had that condition in the definition of du-path, prime path coverage is also simple paths. So du-paths are simple paths. So prime path coverage, surprisingly, subsumes all du-paths coverage. So prime path coverage is a powerful coverage criteria.

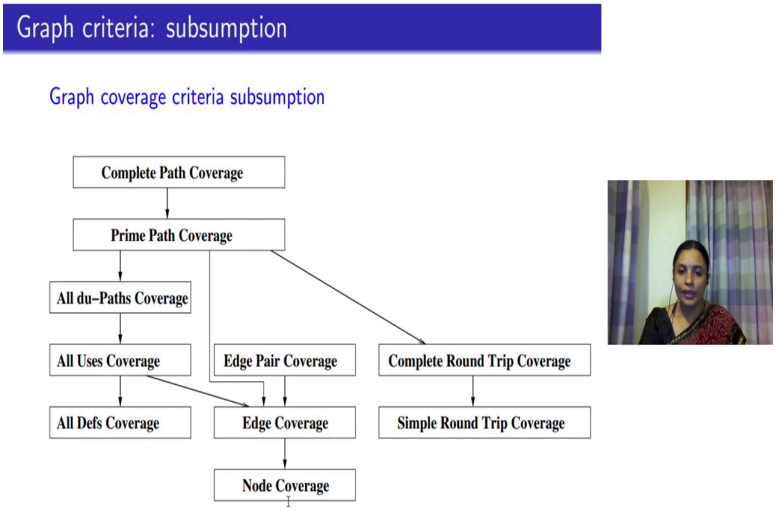
(Refer Slide Time: 21:09)





So what I have basically done in this figure is put the 2 graphs together, the 2 diagrams together. So let us see what it is. Let me just go back. And to start with, we had this structural coverage criteria purely based on structure. Keep this diagram the subsumption that you see in this slide in your mind. Then this lecture, we introduced data flow coverage criteria, which had this obvious subsumption relation, and then we made a few observations, how these 2 compare.

(Refer Slide Time: 21:41)



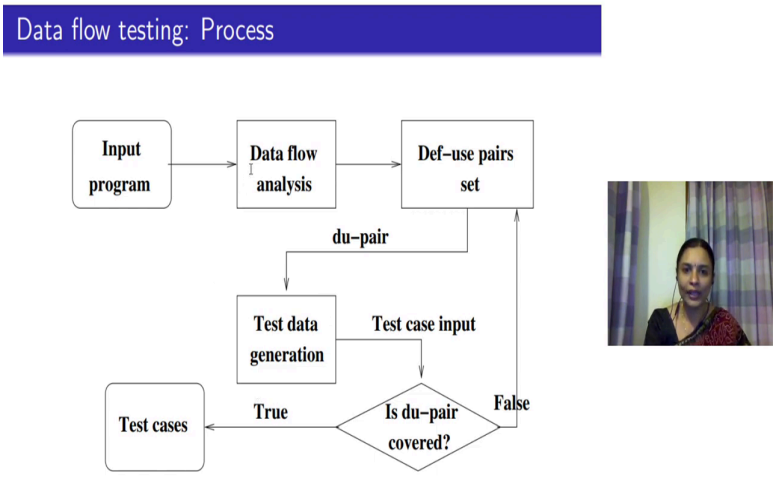
This diagram summarizes. So it says prime path coverage here. subsumes all du-paths coverage. Is this clear? because why? Simple assumption that every du-path is a simple path, prime path are maximal simple path, so it is subsumed. Independently, we saw all uses

coverage, subsumes edge coverage. Is this clear? So that is this new arrow that you see. And this one is this new arrow that you see.

Otherwise, these 3 blocks here are the data flow coverage criteria. The rest of these coverage criteria are structural coverage criteria. I have just put them together with their new observation about what subsumes want. So this is a summary slide on how the various graph coverage criteria that we saw till now, structural and data flow, what subsumes want.

So from this picture, it is very clear that if I am able to draw the control flow graph, and achieve prime path coverage as a test requirement, then I would have successfully done several other coverage criteria. It is a very expressive coverage criteria. Sometimes it might get difficult to enumerate all prime paths. So we might be done with doing smaller coverage criteria, like all uses, or edge coverage, or all du-paths and things like that. But if you can do prime path coverage, then basically you have subsumed several other coverage criteria. That is what it means.

(Refer Slide Time: 23:23)

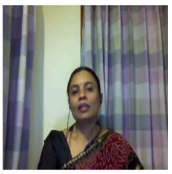


So data flow testing is actually a lot of work. because you need the program, you need to generate the CFG, then you need to do this process called data flow analysis. We may not be doing too much of data flow analysis in this course. I teach you a bit about it when we move later, because without doing the data flow analysis, for large pieces of program, it is difficult to take the control flow graph and augment it with definitions and uses of variables and get this du pair set automatically.

You need some amount of automation processes to be able to take the input program, generate the control flow graph, augment it with definitions and uses and get your du pairs. Once you have your du pairs to get the test data generation also need some work. And then you can measure whether coverage criteria is satisfied or not and get your test cases. You may not look at precise algorithms for this because these will involve sophisticated program analysis techniques.

(Refer Slide Time: 24:29)

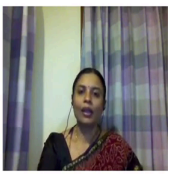
- There are several algorithms for data flow testing that cover each of the stages outlined in the previous slide.
- Identifying du-pairs use program analysis tools, there are several research papers in this area.
  - The number of du-pairs can be very large.
  - Some of the du-pairs can be infeasible. Identifying infeasible du-pairs is usually undecidable.
- Several approaches for test data generation exist:
  - Explicit search, random testing, symbolic execution, model checking etc.



But there are several algorithms for generating du pairs and for generating test data generation. You can do explicit search, which is what I illustrated to you, you can get test data using random testing. You will see a bit of this later in the course. You can get the test data using symbolic execution. Symbolic execution is a very powerful testing technique. So we will spend a lot of time later in this course you can also use another technique called model checking. So these various techniques one after the other, we will see a subset of them, especially random testing and symbolic execution that will help you to get data flow test cases.

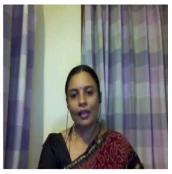
(Refer Slide Time: 25:09)

- There are several algorithms for data flow testing that cover each of the stages outlined in the previous slide.
- Identifying du-pairs use program analysis tools, there are several research papers in this area.
  - The number of du-pairs can be very large.
  - Some of the du-pairs can be infeasible. Identifying infeasible du-pairs is usually undecidable.
- Several approaches for test data generation exist:
  - Explicit search, random testing, symbolic execution, model checking etc.



Credits

Part of the material used in these slides are derived from the presentations of the book Introduction to Software Testing, by Paul Ammann and Jeff Offutt.



If you would like to know a little more on how data flow coverage criteria has worked, here are some reference papers that you can study. They are very readable. This is a survey paper so it has a lot of references. So I will stop here. We will continue with data flow testing in the next class.