



Dependable Software Systems

Topics in Program Slicing

Material drawn from [Weiser84,Gallagher91,DeMillo96]
Courtesy Spiros Mancoridis

Dependable Software Systems (Slicing)



What is a Program Slice?

- A program slice is a subset of a program.
- Program slicing enables programmers to view subsets of a program by filtering out code that is not relevant to the computation of interest.
- *E.g.*, if a program computes many things, including the average of a set of numbers, slicing can be used to isolate the code that computes the average.



Why is Program Slicing Useful?

- Program slices are more manageable for testing and debugging.
- When testing, debugging, or understanding a program, most of the code in the program is irrelevant to what you are interested in.
- Program slicing provides a convenient way of filtering out “*irrelevant*” code.
- Program slices can be computed automatically by statically analyzing the data and control flow of the program.



Definition of Program Slice

- Assume that:
 - P is a program.
 - V is the set of variables at a program location (line number) n .
- A slice $S(V, n)$ produces the portions of the program that contribute to the value of V just before the statement at location n is executed.
- $S(V, n)$ is called the *slicing criteria*.



A Program Slice Must Satisfy the Following Conditions:

- Slice $S(V, n)$ must be derived from P by deleting statements from P .
- Slice $S(V, n)$ must be syntactically correct.
- For all executions of P , the value of V in the execution of $S(V, n)$ just before the location n must be the same value of V in the execution of the program P just before location n .



Example:

Assume the Following Program ...

```
main() {
1.  int mx, mn, av;
2.  int tmp, sum, num;
3.
4.  tmp = readInt();
5.  mx = tmp;
6.  mn = tmp;
7.  sum = tmp;
8.  num = 1;
9.
10. while(tmp >= 0)
11.  {
12.  if (mx < tmp)
13.    mx = tmp;
14.  if (mn > tmp)
15.    mn = tmp;
16.  sum += tmp;
17.  ++num;
18.  tmp = readInt();
19.  }
20.
21. av = sum / num;
22. printf("\nMax=%d", mx);
23. printf("\nMin=%d", mn);
24. printf("\nAvg=%d", av);
25. printf("\nSum=%d", sum);
26. printf("\nNum=%d", num);
}
```



Slice S(num, 26)

```
main() {  
2.  int tmp, num;  
4.  tmp = readInt();  
8.  num = 1;  
10. while(tmp >= 0)  
11.  {  
17.  ++num;  
18.  tmp = readInt();  
19.  }  
26. printf("\nNum=%d", num);  
}
```



Slice S(sum, 25)

```
main() {  
2.  int tmp, sum;  
4.  tmp = readInt();  
7.  sum = tmp;  
10. while(tmp >= 0)  
11.  {  
16.    sum += tmp;  
18.    tmp = readInt();  
19.  }  
25. printf("\nSum=%d", sum);  
}
```



Slice S(av, 24)

```
main() {  
1.  int av;  
2.  int tmp, sum, num;  
4.  tmp = readInt();  
7.  sum = tmp;  
8.  num = 1;  
10. while(tmp >= 0)  
11.  {  
16.  sum += tmp;  
17.  ++num;  
18.  tmp = readInt();  
19.  }  
21. av = sum / num;  
24. printf("\nAvg=%d", av);  
}
```



Slice S(mn, 23)

```
main() {  
1. int mn;  
2. int tmp;  
4. tmp = readInt();  
6. mn = tmp;  
10. while(tmp >= 0)  
11. {  
14. if (mn > tmp)  
15.     mn = tmp;  
18. tmp = readInt();  
19. }  
23. printf("\nMin=%d", mn);  
}
```



Slice $S(mx, 22)$

```
main() {  
1. int mx;  
2. int tmp;  
4. tmp = readInt();  
5. mx = tmp;  
10. while(tmp >= 0)  
11. {  
12.   if (mx < tmp)  
13.     mx = tmp;  
18.   tmp = readInt();  
19. }  
22. printf("\nMax=%d", mx);  
}
```



Observations about Program Slicing

- Given a slice $S(X, n)$ where variable X depends on variable Y with respect to location n :
 - All **d-uses** and **p-uses** of Y before n are included in $S(X, n)$.
 - The **c-uses** of Y will have no effect on X unless X is a **d-use** in that statement.
- Slices can be made on a variable at any location.



Program Slicing Process

- Select the slicing criteria (*i.e.*, a variable or a set of variables and a program location).
- Generate the program slice(s).
- Perform testing and debugging on the slice(s). During this step a sliced program may be modified.
- Merge the modified slice with the rest of the modified slices back into the original program.



Tools for Program Slicing

- **Spyder**
 - A debugging tool based on program slicing.
 - **<ftp://bagdemagus.cs.purdue.edu/Spyder/>**
- **Unravel**
 - A program slicer for ANSI C.
 - **<ftp://hissa.ncsl.nist.gov/unravel>**



References

- [Weiser84] Weiser, M., *Program Slicing*, IEEE Transactions on Software Engineering, Vol. SE-10, No. 4, July, 1984.
- [Gallagher91] Gallagher, K. B., Lyle, R. L., *Using Program Slicing in Software Maintenance*, IEEE Transactions on Software Engineering, Vol. SE-17, No. 8, August, 1991.
- [DeMillo96] DeMillo, R. A., Pan, H., Spafford, E. H., *Critical Slicing for Software Fault Localization*, Proc. 1996 International Symposium on Software Testing and Analysis (ISSTA), San Diego, CA, January, 1996.