Slicing

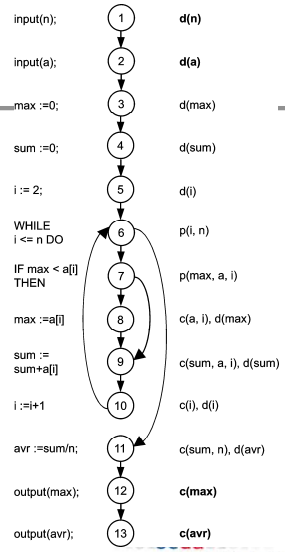
Content

- Characteristics and Goals
- Static vs. Dynamic Slicing
- Static Slicing
- Dynamic Slicing

Characteristics and Goals

- Slicing is related to the extensively automatic provision of information about interactions in a program under observation. The operation mode is comparable to manual fault finding (debugging) after the occurrence of a failure. Debugging is also a typical application area for automatic slicing.
- A program slice in this context describes which instruction affects an observed value in which way. A slice will always be given in relation to an observed value at a certain position of the program. In a strict sense, this is the definition of so-called backward slicing. There is also forward slicing. Such a slicing shows which instruction is affected by an observed value in which way.

Slicing
Exemplar Module

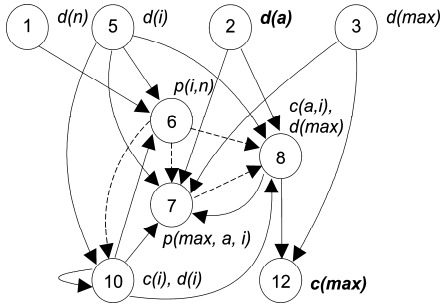


Static vs. Dynamic Slicing

- There exist static und dynamic slicing.
- **Static slicing** can be executed in the sense of static analysis. It is not necessary to execute the program or to make assumptions about the values of variables. This is critical if dependencies only result from the concrete value of a datum. Such situations exist, e.g., in conjunction with the application of pointers or fields.
- A remedy is offered by so-called **dynamic slicing**.

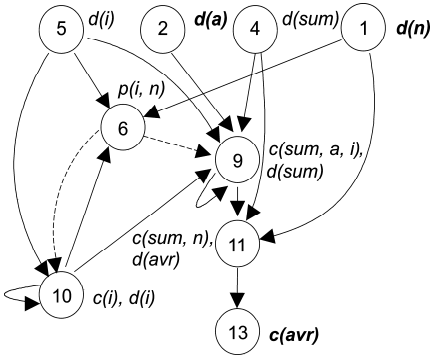
Static Slicing

- Slice to *max*



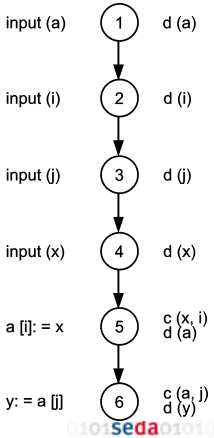
Static Slicing

- Slice to *avr*



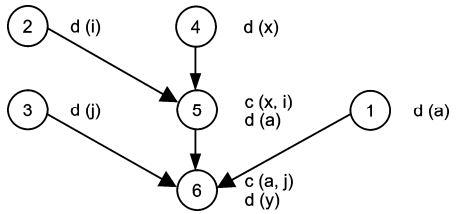
Dynamic Slicing

- A common drawback of all static procedures is that the information that is first generated at the execution time cannot be considered completely.



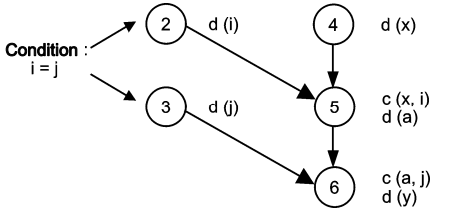
Dynamic Slicing

- Static slice for y in node 6 (is too complicated for every concrete situation)



Dynamic Slicing

- Dynamic slice for y in node 6 with i = j



Dynamic Slicing

- Dynamic slice for y in node 6 with i not equaling j