Backward Static Program Slicing

Kevin Coogan
Math 543
12/8/08

Program Slicing

- *slice* – an executable program that is obtained from the original program by deleting zero or more statements.

Program Slicing

- *static slice criterion* – consists of a pair (*n,* V) where *n* is a node in the Control Flow Graph, and V is a subset of the program's variables.

- *dynamic slice criterion* – consists of a triple (n, V, I) where I is an input to the program.

- *Forward* – subset of instructions effected by criterion.

- *Backward* – subset of instructions that contribute to value of variables in criterion.

Backward Static Slicing -- Initialization

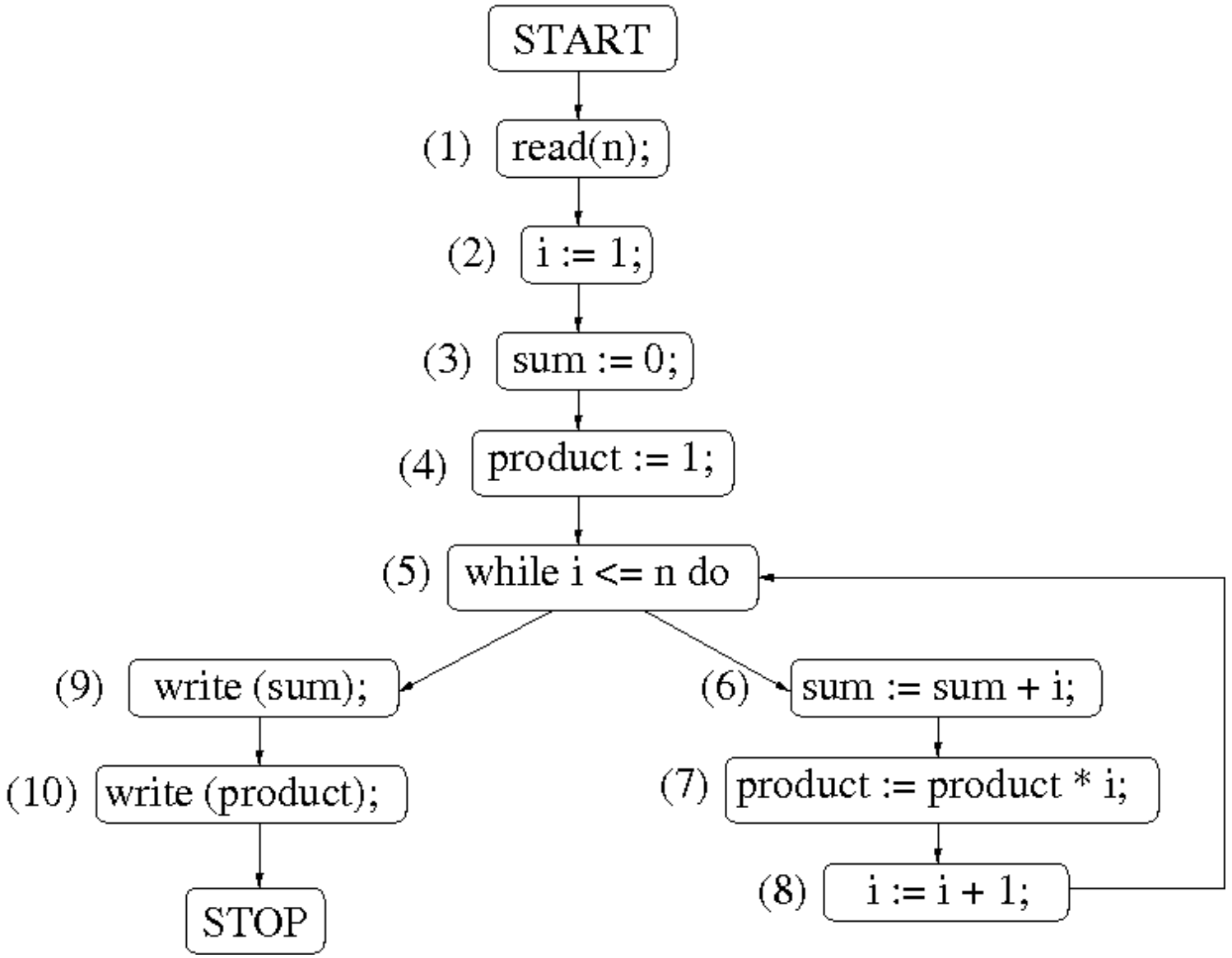
- Build Control Flow Graph of the program.
- Set relevant variables of criterion statement to V.
- Set relevant variables for all other statements to empty set.

Sample Program

```
read(n);
i := 1;
sum := 0;
product := 1;
while i <= n do
begin
    sum := sum + i;
    product := product * i;
    i := i + 1;
end;
write (sum);
write (product);
```

- criterion = (10, product)

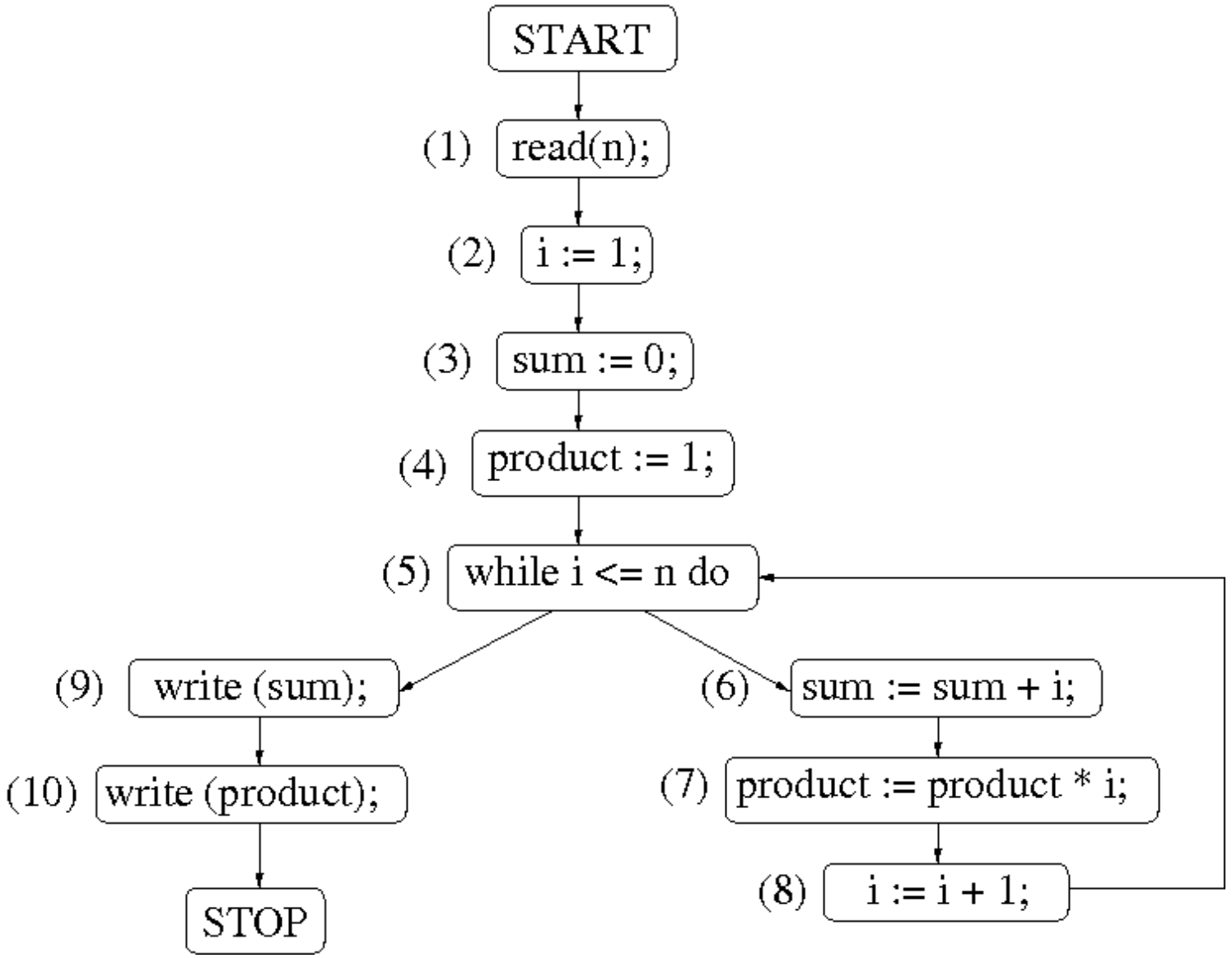
Control Flow Graph



Backward Static Slicing Directly Relevant Variables

- For each edge i-j in the CFG:
  - if i does not define relevant variable of j, add that variable to i’s relevant variables.
  - if i does define relevant variable of j, add variables referenced by i to i’s relevant variables.
  - In presence of loops, need to iterate until no change.

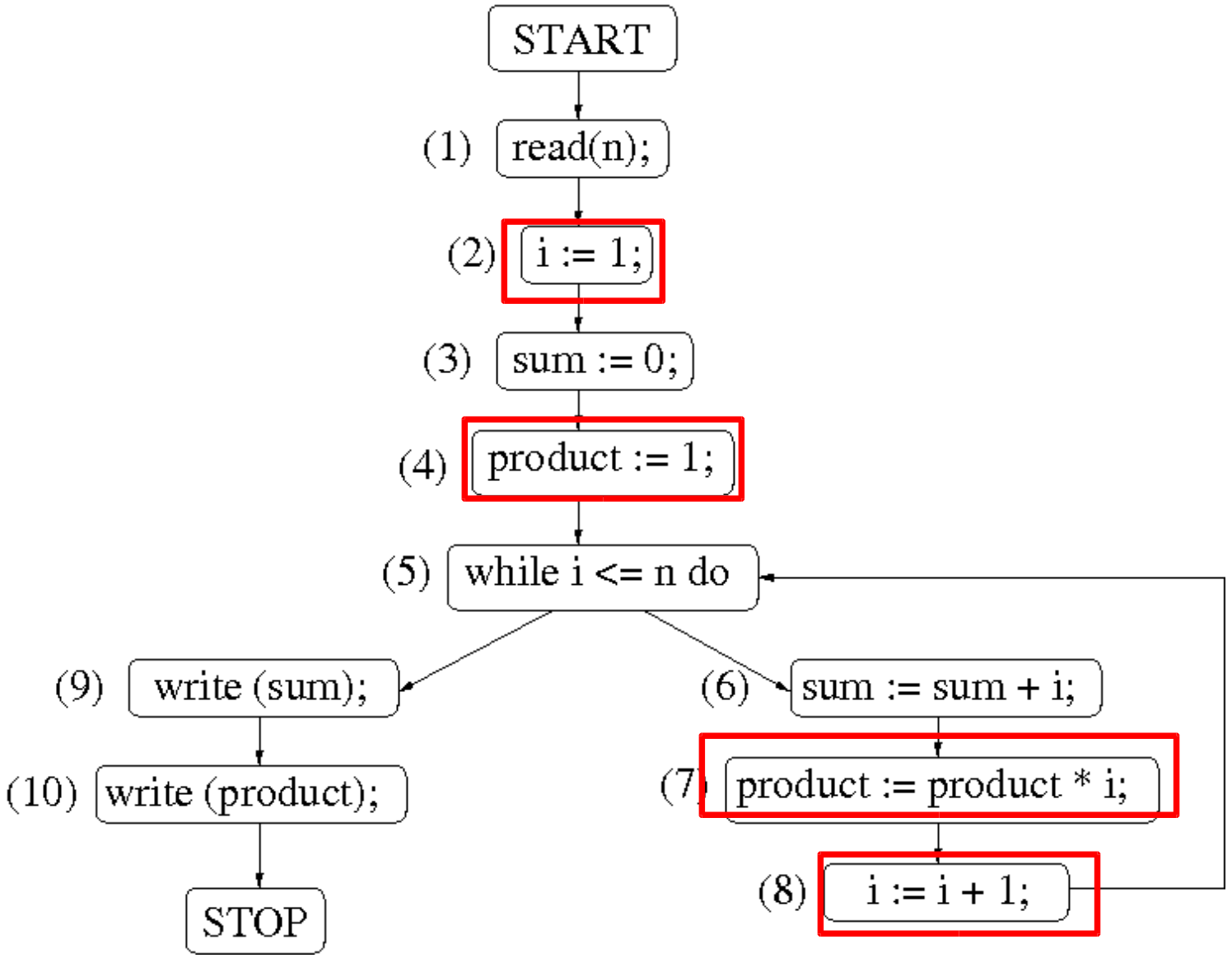
Control Flow Graph



Backward Static Slicing
Slice Statements

- For each edge i-j in the CFG:
  - if i defines a relevant variable of j, add i to slice.

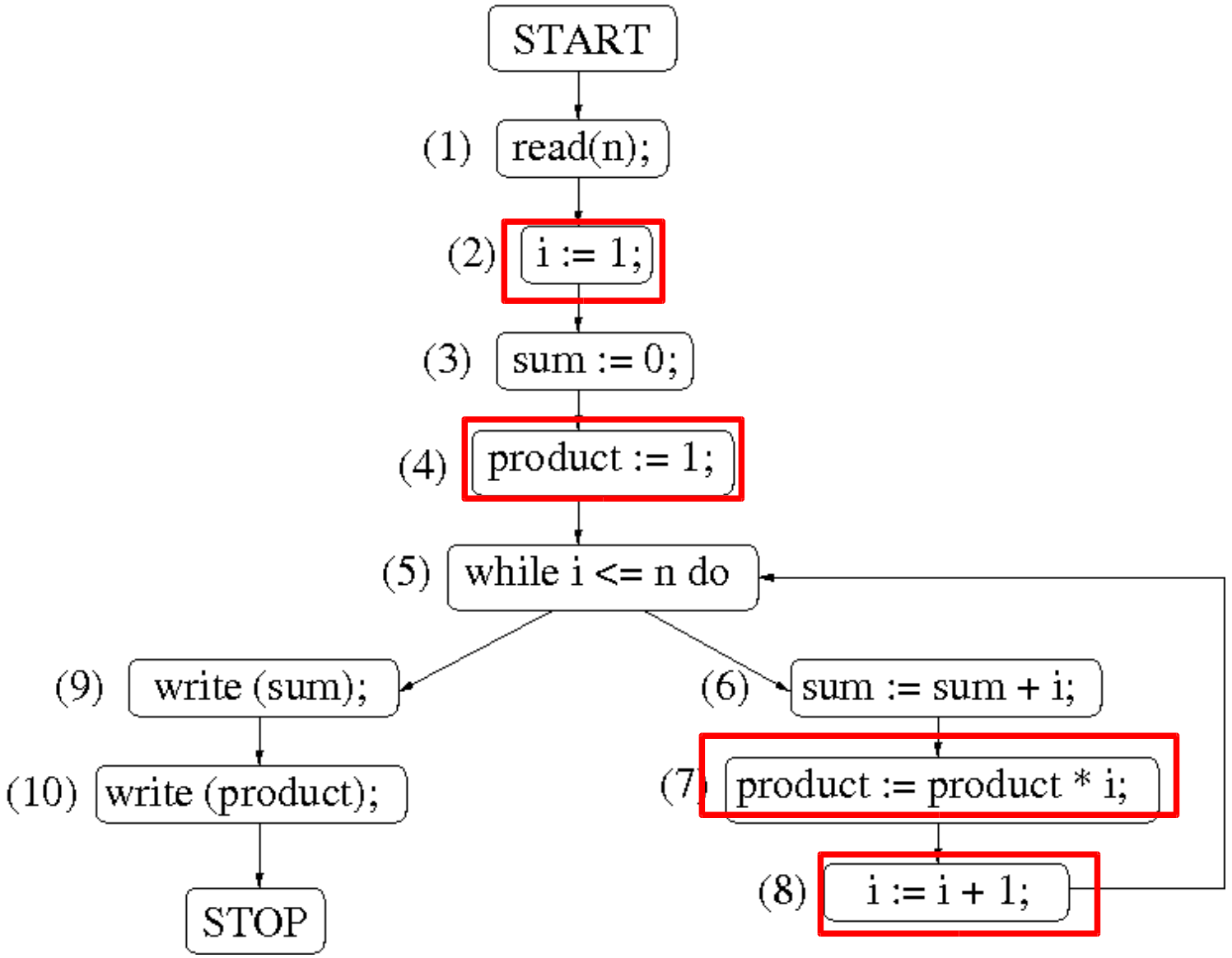
Control Flow Graph



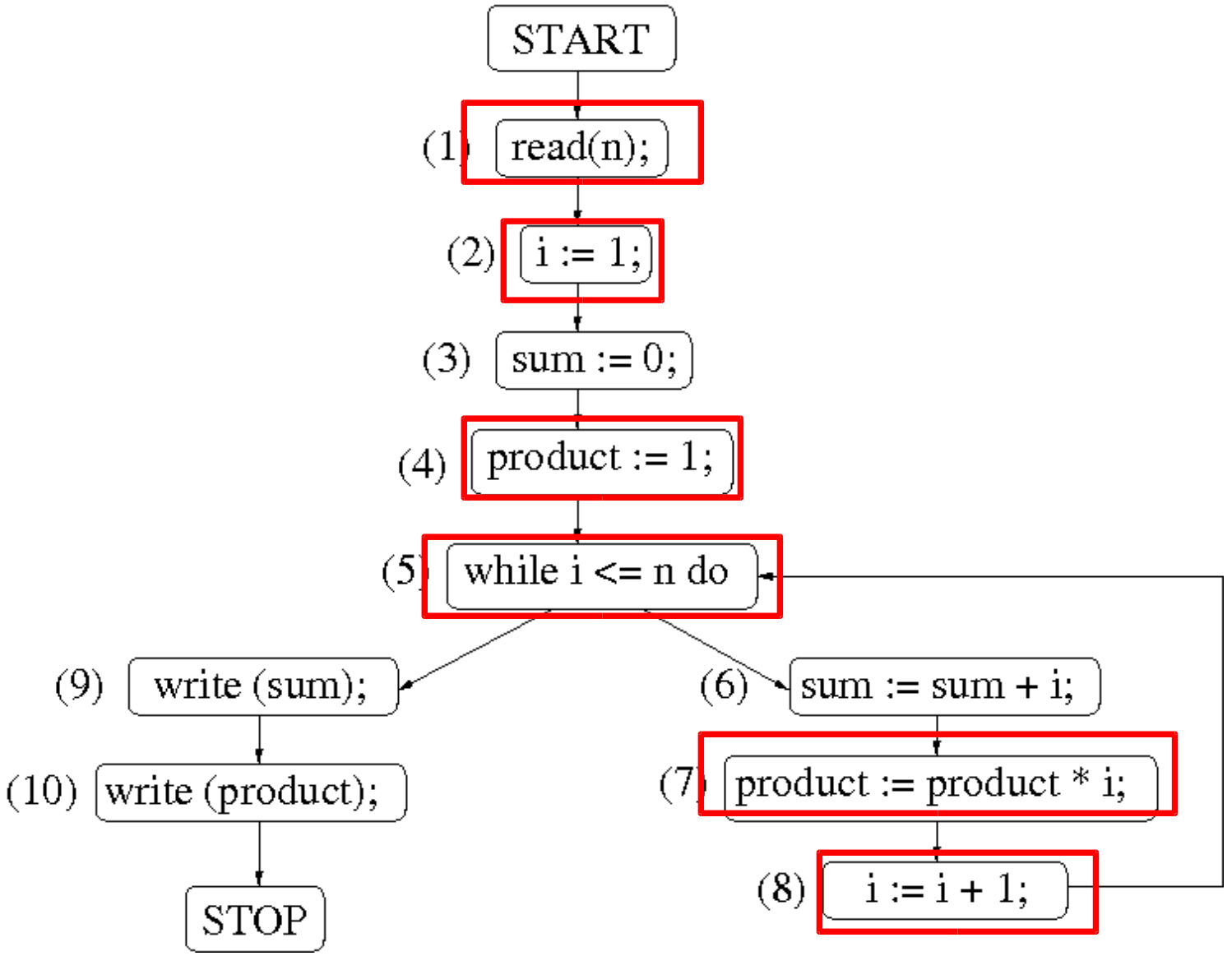
Backward Static Slicing Indirectly Relevant Variables

- For each branch statement *b* in the CFG:
  - if an i exists such that i is in slice, and i is control dependent on b, then
  - add *b* to slice
- calculate directly relevant variables of criterion (b, Ref(b))
- Identify slice instructions as before

Control Flow Graph



Control Flow Graph



Sample Program

```
read(n);
i := 1;
sum := 0;
product := 1;
while i <= n do
begin
    sum := sum + i;
    product := product * i;
    i := i + 1;
end;
write (sum);
write (product);
```

```
read(n);
i := 1;

product := 1;
while i <= n do
begin

    product := product * i;
    i := i + 1;
end;
```

Questions ? ? ?