Software Testing and Reliability

Xiaoyuan Xie 谢晓园
xxie@whu.edu.cn
计算机学院E301

Lecture 11

Program Analysis
(Data Flow Analysis)

Data Flow Analysis

- NOT the same as the data flow coverage testing
- A testing method for detecting improper use of variables in programs

  -- “Improper use of a variable” = “improper sequence of actions on a variable”

Data Flow Analysis (continued)

- Three possible actions

  -- Define - d

    -- The variable is assigned a value

  -- Reference - r

    -- The variable’s value is referred

  -- Undefine - u

    -- The variable is declared but not yet assigned any value

    -- The variable’s value is destroyed

    -- The variable’s value goes out of the scope

Data Flow Analysis (continued)

- **Data Flow Anomalies**

  -- **undefine-reference (called ur anomaly)**

  -- A variable has not been assigned any value, but its value is referred in the program

  -- **define-define (called dd anomaly)**

  -- A variable's value has been defined but not used before another value is assigned to it

  -- **define-undefine (called du anomaly)**

  -- A variable's value has been defined but not used before the value is destroyed

Dynamic Data Flow Analysis

- **Make use of *program instrumentation***

  -- Insert statements into the program under analysis

  -- Execute the *instrumented program* with input data

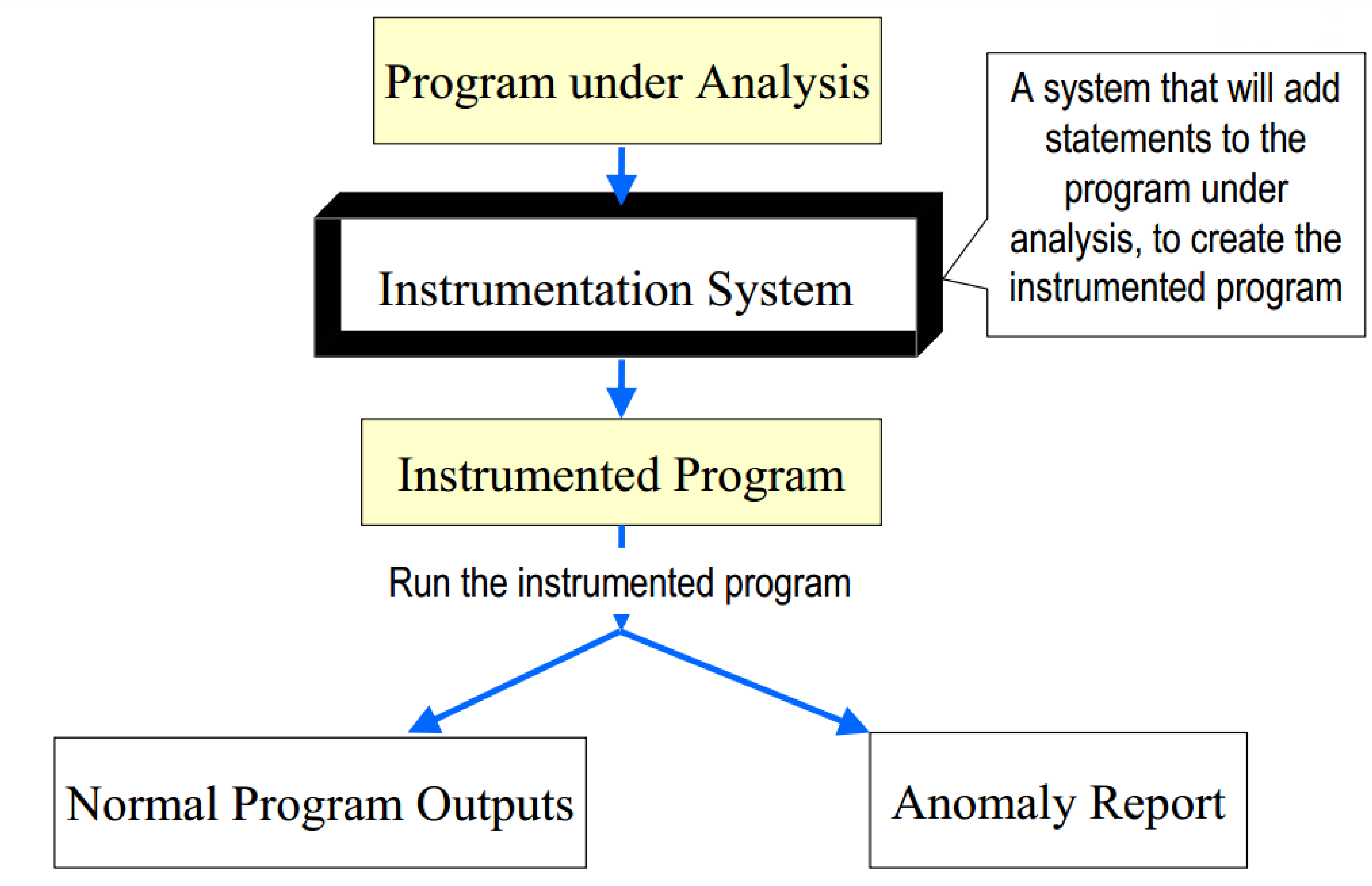
    -- Gather some run time information. For example, find out whether v[h] and v[k] are the same variable or not

    -- Keep track of the actions on each variable

    -- Change the state of the right variable at the right place

- Identify anomalies by keeping track of state transitions

Dynamic Data Flow Analysis



Anomaly Report

- Anomaly message consists of

  -- Variable name

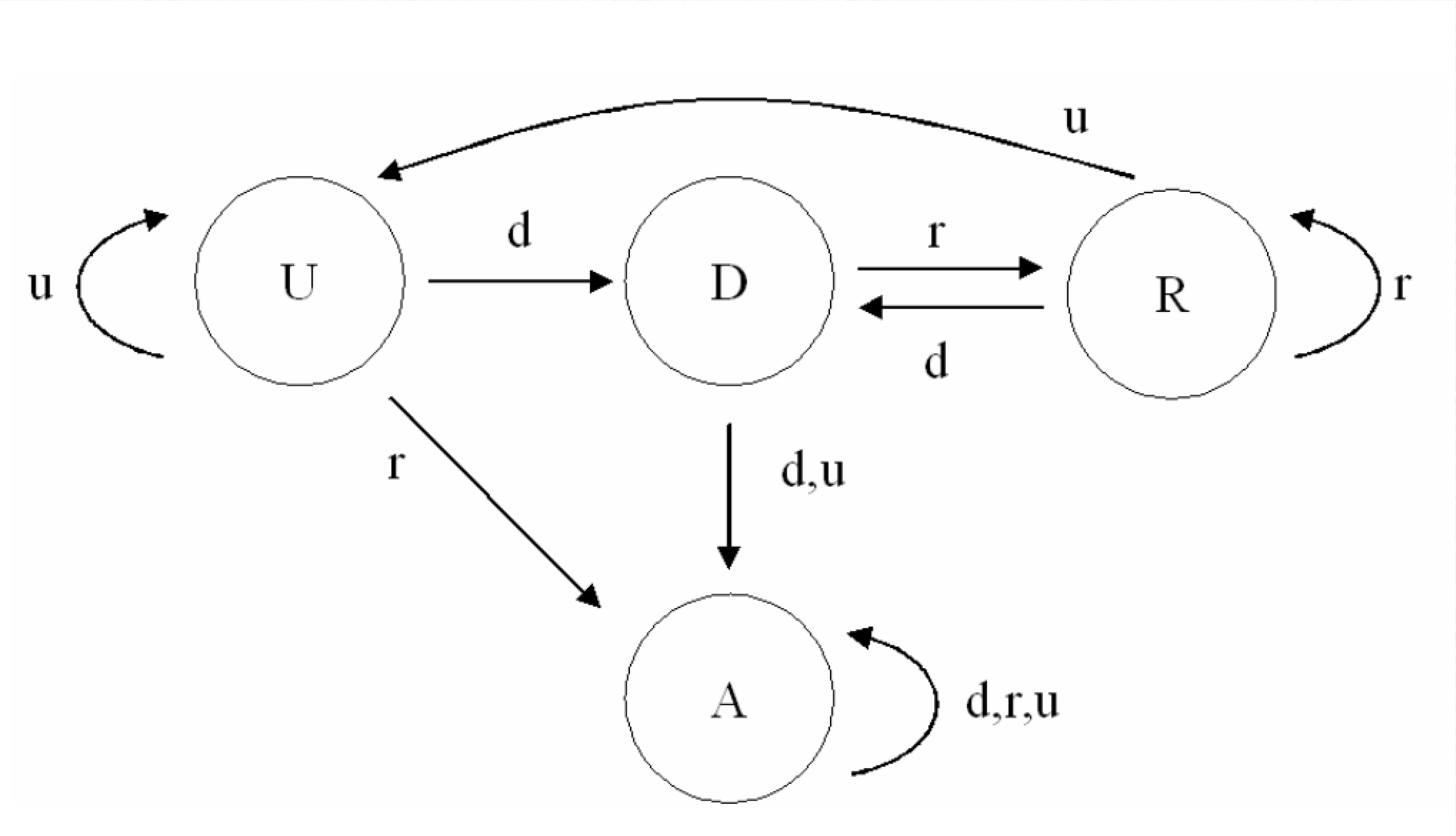
  -- Types of anomalies

  -- Positions of the pair of actions giving rise to the anomaly

Types of States

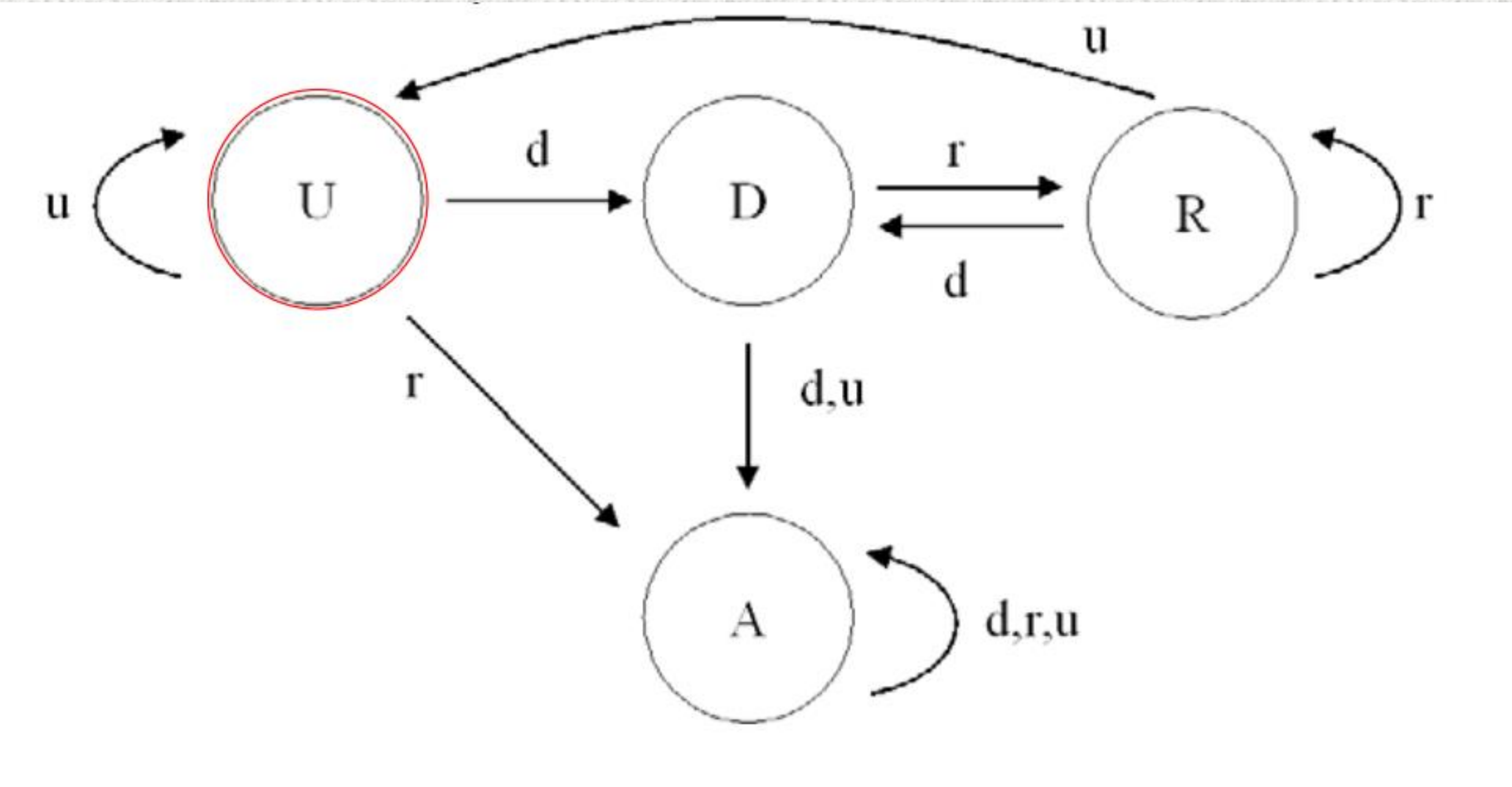
- State U : undefined
- State D : defined
- State R : referenced
- State A : abnormal

State Transition Diagram 1



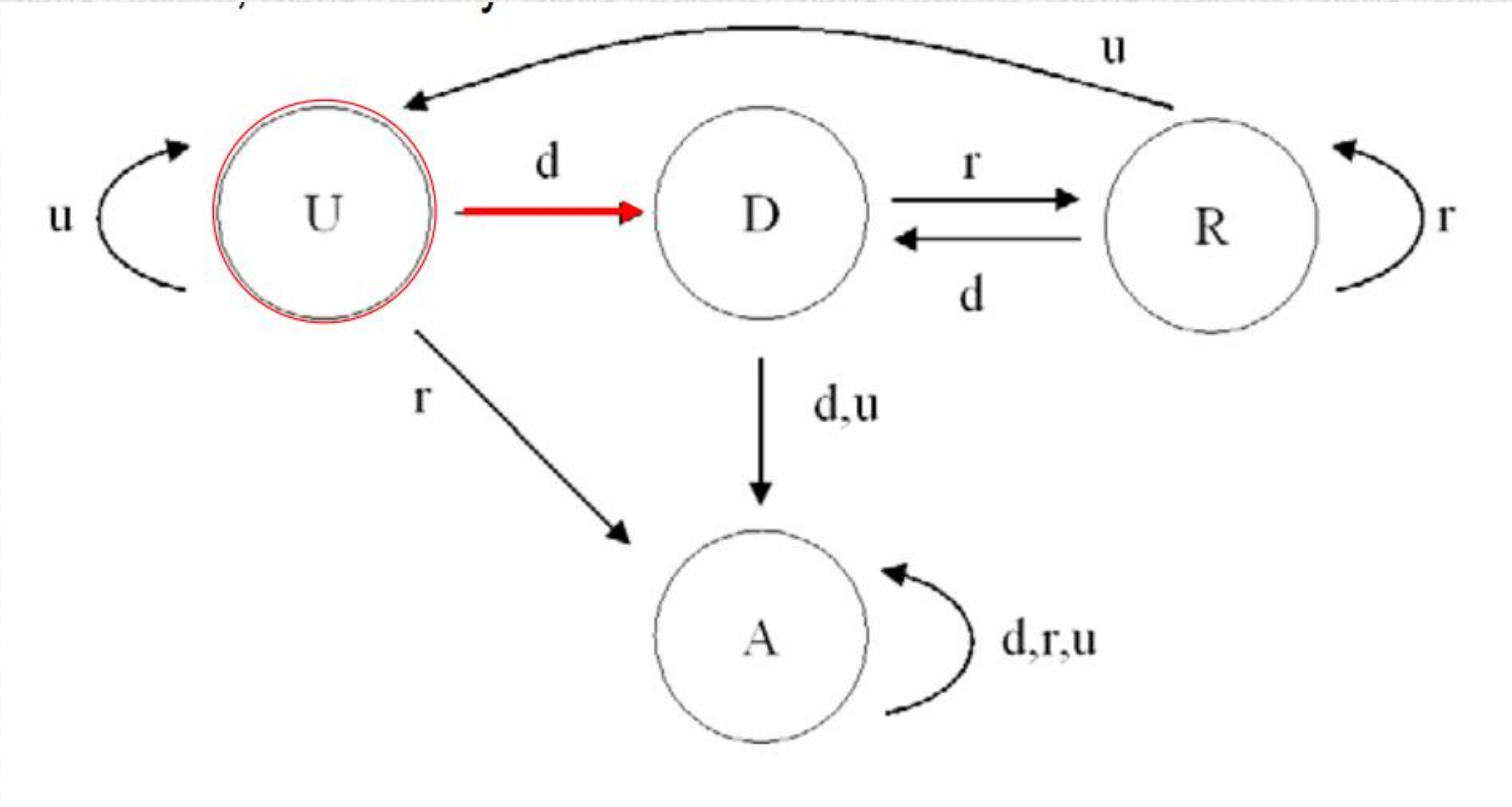
State Transition during Testing

Suppose that the initial state of x is U, and the define action occurs, followed by reference.



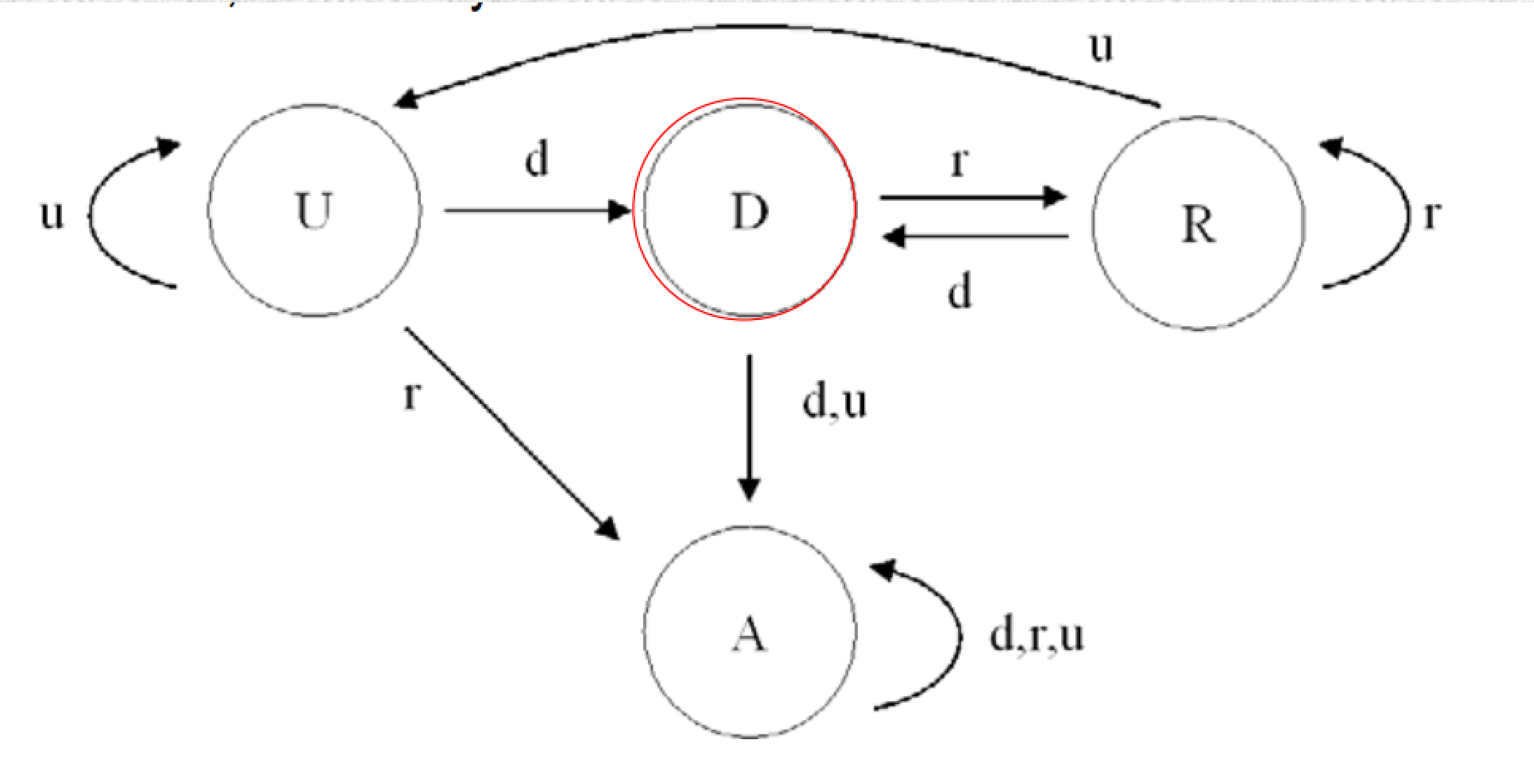
State Transition during Testing (continued)

Suppose that the initial state of x is U, and the *define* action occurs, followed by *reference*.



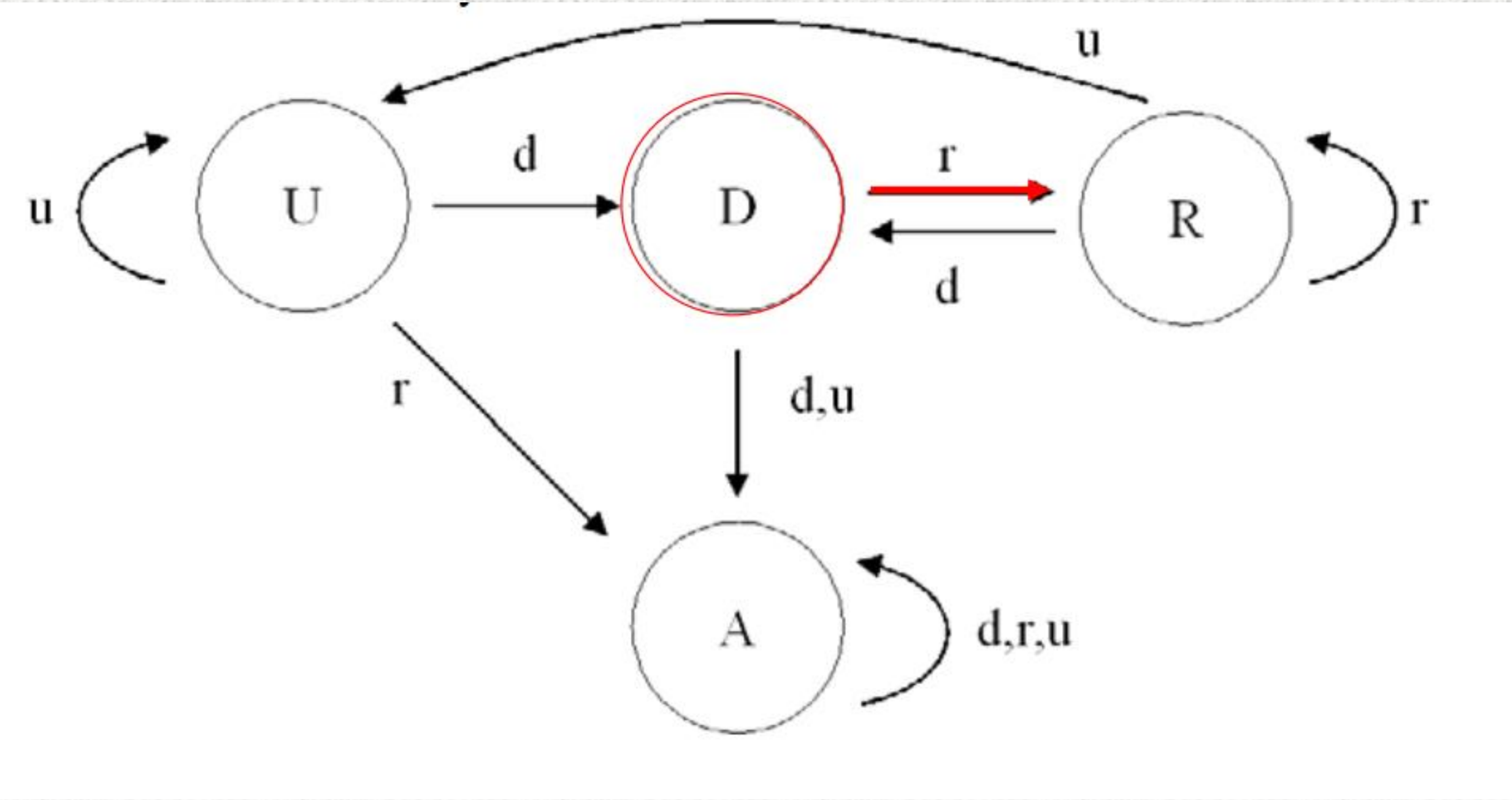
State Transition during Testing (continued)

Suppose that the initial state of x is U, and the *define* action occurs, followed by *reference*.



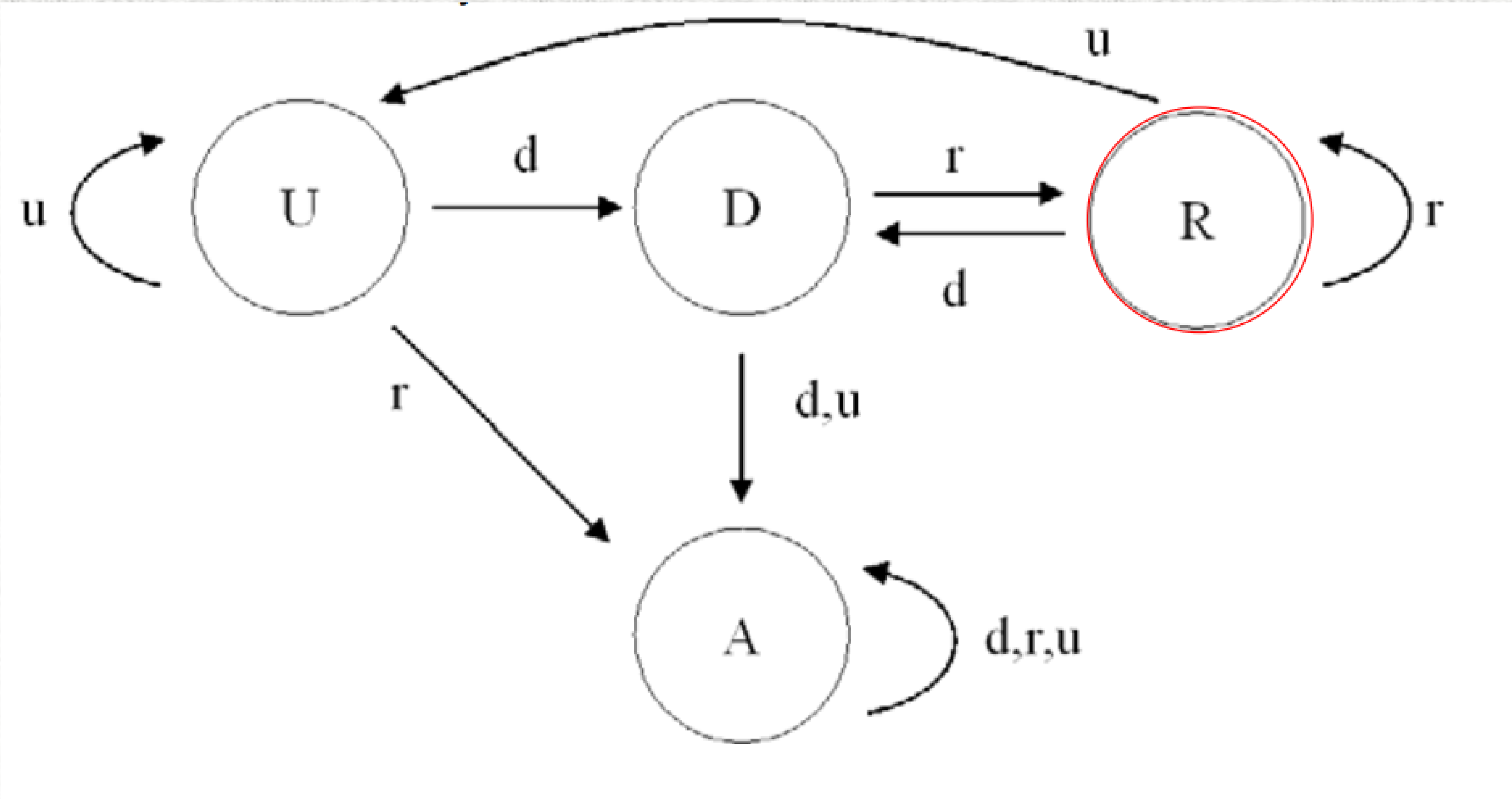
State Transition during Testing (continued)

Suppose that the initial state of x is U, and the *define* action occurs, followed by *reference*.



State Transition during Testing (continued)

Suppose that the initial state of x is U, and the *define* action occurs, followed by *reference*.



State Variable

- For each variable (say, A) under analysis, we need to create another variable to store the state of A.
  - -- This another variable is called the “state variable” of A

State Transition Function

```
next-state f (current-state, action)
{ if (current-state = U and action = d) then return D;
  if (current-state = D and action = d ) then return A;
  ........
  ........
  if (current-state = A) then return A;
  end
}
```

State Transition Function (continued)

Examples of calling the state transition function

$$\underline{f}\ (\underline{\text{state-of-x}}, \underline{rd}\ ) \equiv f\ (f(\text{state-of-x}, \underline{r}), \underline{d})$$

Call the state transition function

$$f(\text{state-of-x}, \underline{rud}) \equiv f(f(f(\text{state-of-x}, \underline{r}), \underline{u}), \underline{d})$$

Example 1

Consider the following program segment

```
B = 10.0
Input A
A = B + A
A = B + B
```

Example 1 (continued)

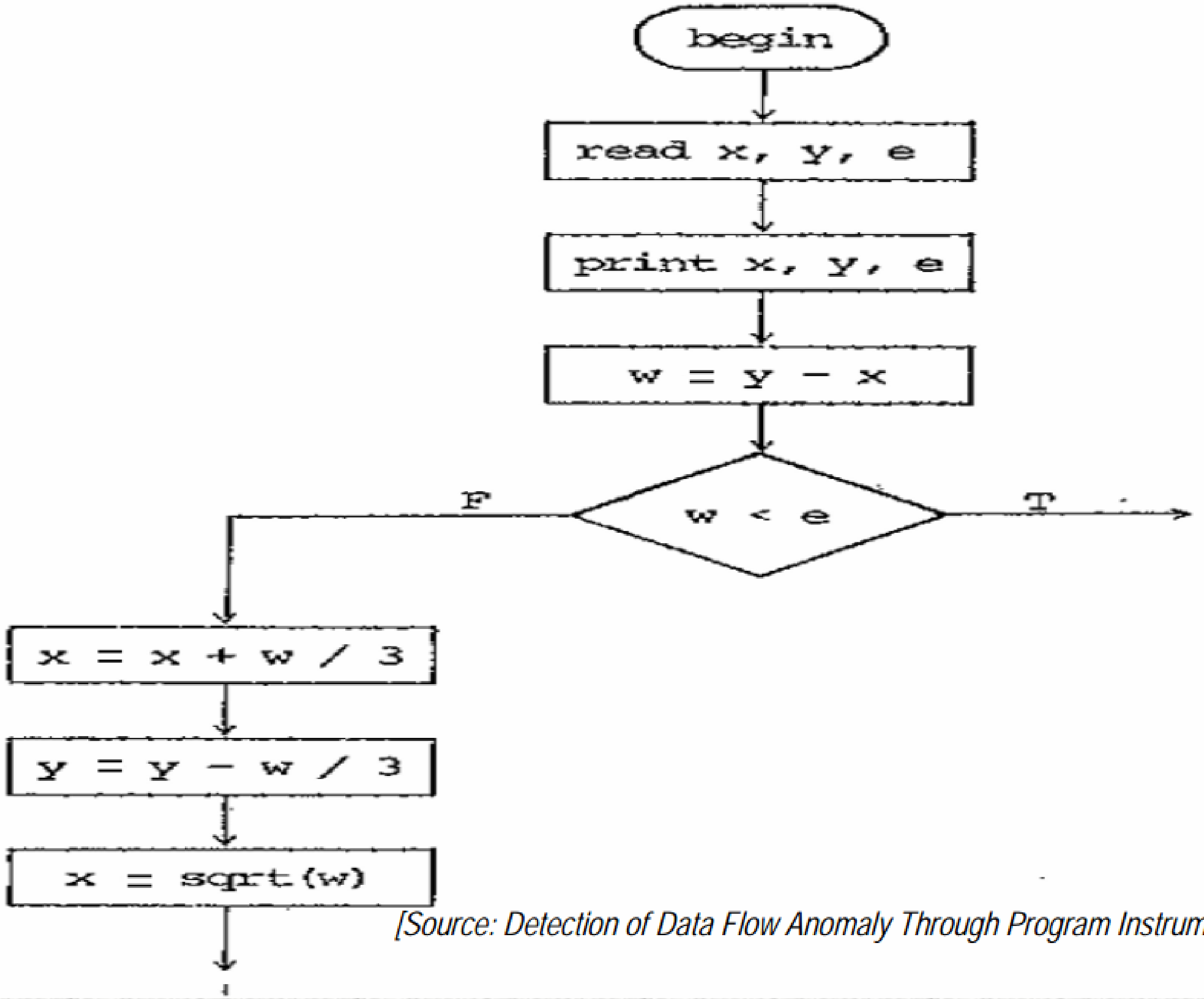
Data flow analysis should examine all the variables. Here, we only demonstrate the analysis on variable A.

Instrumented **program segment**	**Action on A**	**Value of st-A upon execution**
st-A = U		U
B = 10.0		
A = 0	d	
st-A = f(st-A,d)		D
A = B + A	rd	
st-A = f(st-A,rd)		D
A = B + B	d	
st-A = f(st-A,d)		A

This is the **state variable** of "A"

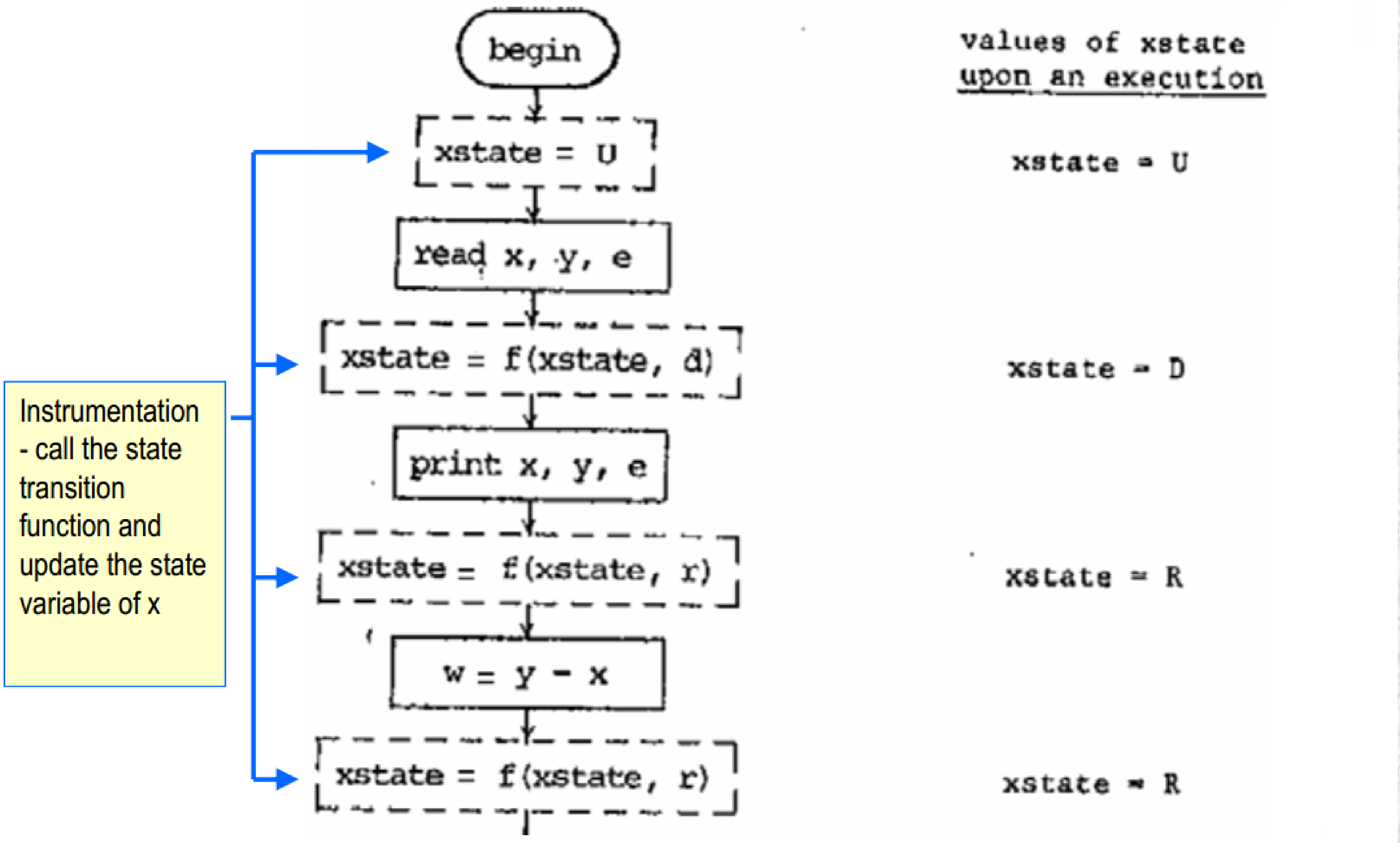
Example 2

Suppose the directed graph of the program under analysis is as follows.



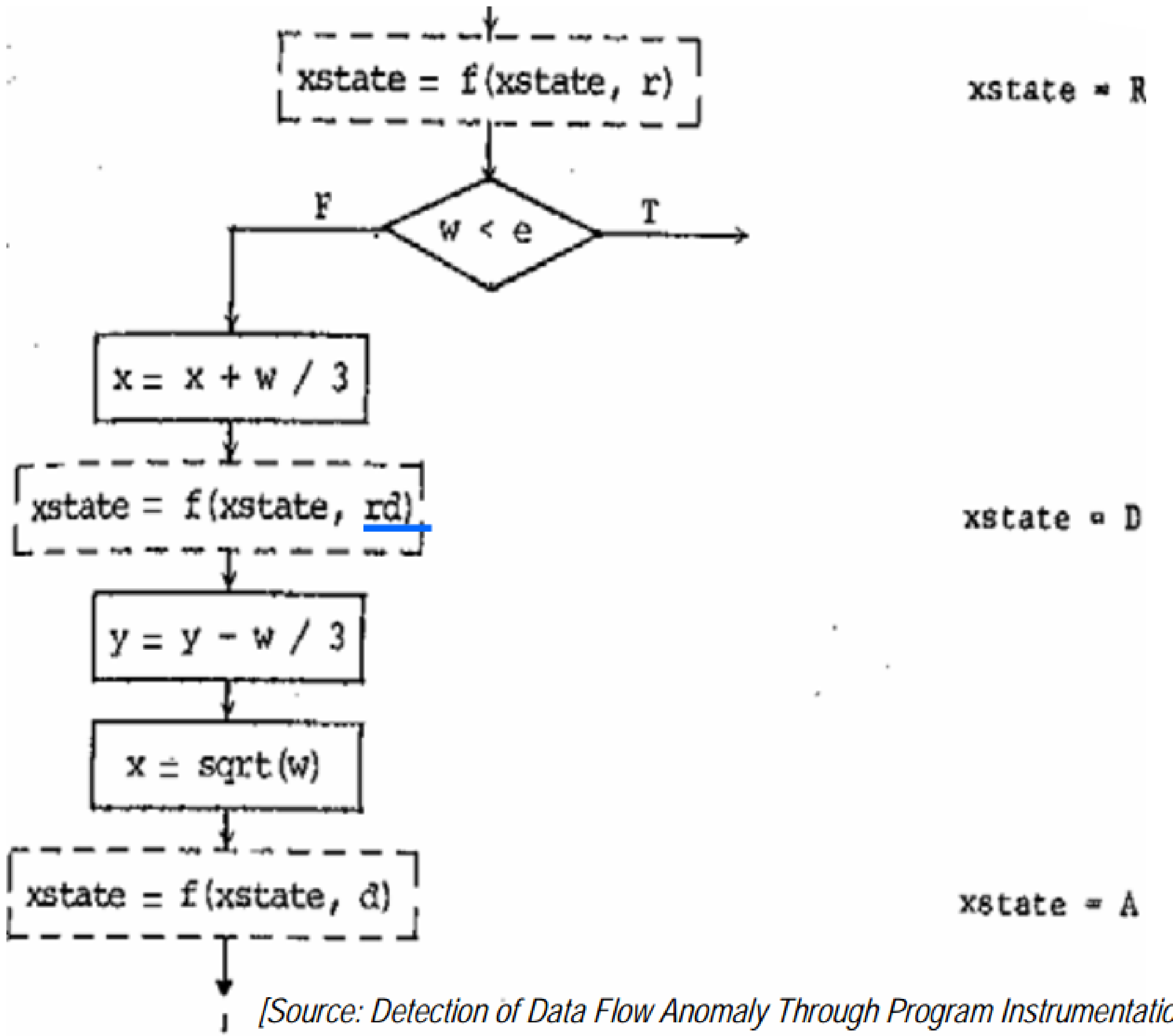
[Source: Detection of Data Flow Anomaly Through Program Instrumentation]

Example 2 (continued)



[Source: Detection of Data Flow Anomaly Through Program Instrumentation]

Example 2 (continued)



[Source: Detection of Data Flow Anomaly Through Program Instrumentation]

Example 3

Instrumentation for the array elements

If the program under analysis contains the following statement

$$\underline{a[i, j]} = \underline{a[i, k]} * \underline{a[k, j]}$$

then the required instrumentation for this statement will be

$$\underline{sta[i, k]} = f(sta[i, k], r);$$

$$\underline{sta[k, j]} = f(sta[k, j], r);$$

and

$$\underline{sta[i, j]} = f(sta[i, j], d).$$

Problems of State Transition Diagram 1

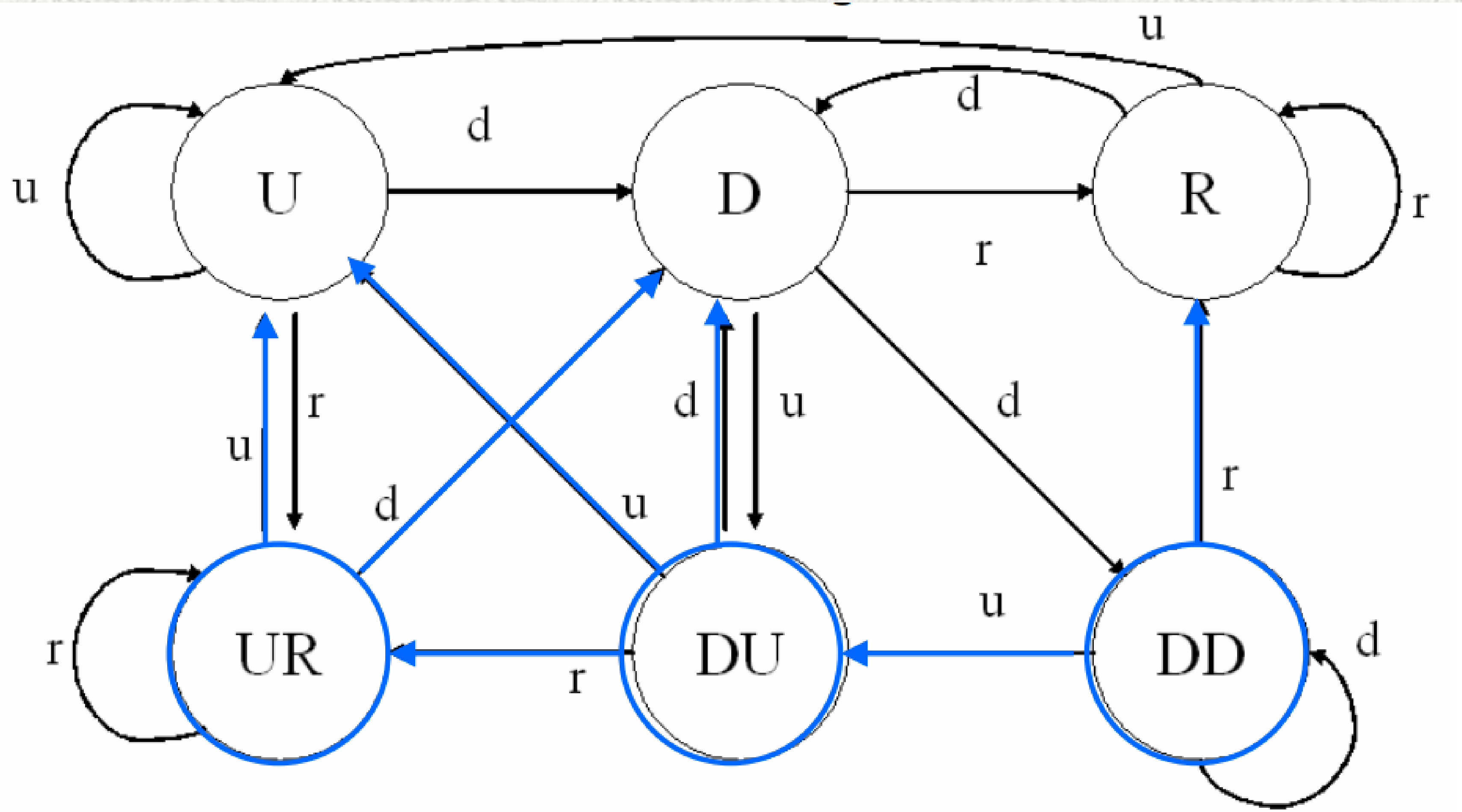
- No distinction between types of anomalies
- Once a variable enters the abnormal state, it will remain there forever
- Two solutions:
  1. Reset the state variable to R
  2. Use an extended state transition diagram

Solution 1

```
new-state f (current-state, action)
{ if (current-state = U and action = d) then return D;
if (current-state = D and action = d ) then report A, but
return R;
......
......
if (current-state = A) then report A, but return R;
end
}
```

Solution 2

An Extended State Transition Diagram



More different states about the anomalies

Blue arrows – Jump out from a *anomaly* state

Solution 2 (continued)

- Change of the state transition diagram only leads to change of the state transition function. It will NOT change the instrumentation.

Exercise

- Instrumentation for loop, and if structure

```
INPUT A, B, C
IF (A=0)
    IF (B=0)
        IF (C=0)
            OUTPUT ("X can be any value.")
        ELSE
            OUTPUT ("There is no solution for X.")
    ELSE
        IF (C=0)
            X = 0
        ELSE
            X = -C/B
ELSE
    IF (B=0)
        IF (C=0)
            X = 0
        ELSE
            IF (A*C<0)
                X = SQRT(-C/A) or X = -SQRT(-C/A)
            ELSE
                OUTPUT ("There is no real solution for X.")
    ELSE
        IF (C=0)
            X = 0 or X = -B/A
        ELSE
            D = B*B-4*A*C
            IF (D>0)
                X = (-B+SQRT(D))/(2*A) or X = (-B-SQRT(D))/(2*A)
            ELSE IF (D=0)
                X = -B/(2*A)
            ELSE
                OUTPUT ("There is no real solution for X.")
```