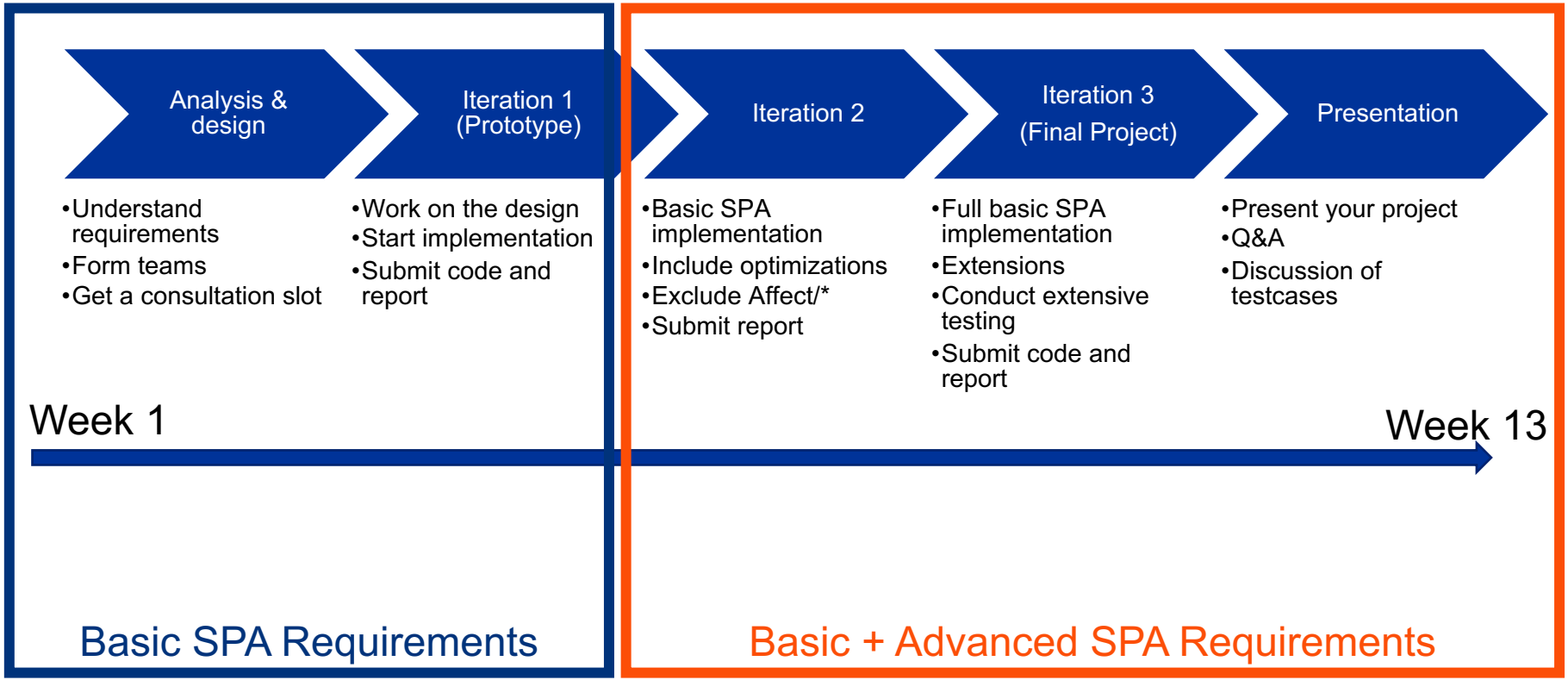
CS3203: Software Engineering Project

Advanced SPA Requirements

By: Dr. Bimlesh Wadhwa

NUS National University of Singapore | School of Computing

CS3203 Project Iterations



Advanced SPA Requirements

- (Advanced or more) design abstractions and design entities
- (Advanced or more) query clauses

More Relationships and Entities

PROGRAM DESIGN ABSTRACTIONS (aka RELATIONSHIPS)

- Calls and Calls*
- Modifies
- Uses
- Follows and Follows*
- Parent and Parent*
- Next and Next*
- Affects and Affects*

PROGRAM DESIGN ENTITIES

- Procedure
- StmtLst
- Stmt
- Assign
- Call
- While
- If
- Variable
- Constant
- Prog_line

More on PQL side

Queries in PQL contain:

- Declaration
  - more types of synonyms
- Select clause specifies query result
  - single or multiple return values (tuples) or BOOLEAN
  - such that clauses constrains the results in terms of relationships
  - pattern clauses contains results in terms of code patterns
  - with clause constrains the results in terms of attribute values
- Query results must check (make true) all “such that”, “with” and “pattern” clauses

More Relationships and Entities

Calls and Calls*

Next and Next*

Affects and Affects*

Source language: SIMPLE

```
procedure First {
    read x;
    read z;
    call Second; }
```

```
procedure Second {
1.      x = 0;
2.      i = 5;
3.      while (i!=0)  {
4.          x = x + 2*y;
5.          call Third;
6.           i = i - 1; }
7.      if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
    z = 5;
    v = z;
    print v; }
```

Calls and Calls*

For any procedures p and q:

- **Calls (p, q)** holds if procedure p directly calls q
- **Calls* (p, q)** holds if procedure p directly or indirectly calls q , that is:
  - Calls (p, q) or
  - Calls (p, p1) and Calls* (p1, q) for some procedure p1

Examples of Calls and Calls*

```
procedure First {
    read x;
    read z;
    call Second; }
```

```
procedure Second {
1.      x = 0;
2.      i = 5;
3.      while (i!=0)  {
4.          x = x + 2*y;
5.          call Third;
6.          i = i - 1; }
7.      if (x==1) then {
8.          x = x+1; }
        else {
9.          z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
     z = 5;
     v = z;
     print v;}
```

Which relationships hold?

A. Calls (“First”, “Second”)

B. Calls (“First”, “Third”)

C. Calls (“Second”, “Third”)

D. Calls (“Second”, “First”)

E. Calls* (“First”, “Second”)

F. Calls* (“First”, “Third”)

G. Calls* (“Second”, “First”)

More Relationships and Entities

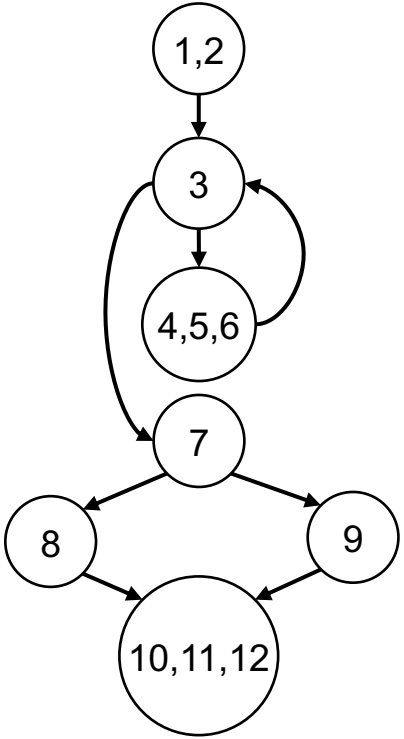
Calls and Calls*

Next and Next*

Affects and Affects*

Control Flow Graph (CFG) *for a procedure*

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.       call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.       x = x+1; }
       else {
9.       z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

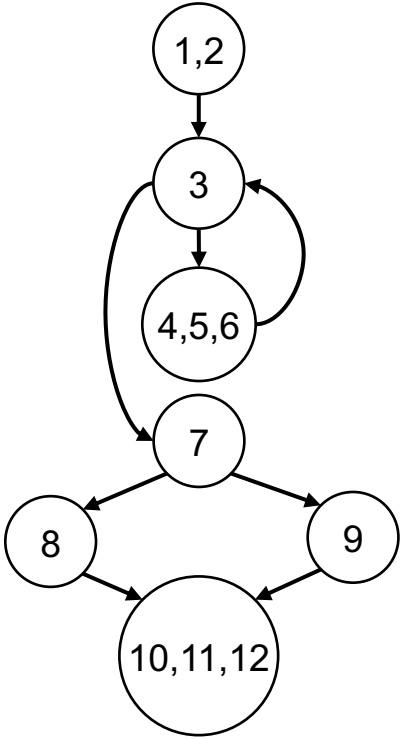


Control Flow Graph - Definition

- Nodes = basic blocks of source code
  - Basic block = a program region with a single entry and single exit point
  - code fragments executed without control transfer
- Directed edges = possibility that program execution proceeds from one basic block to another

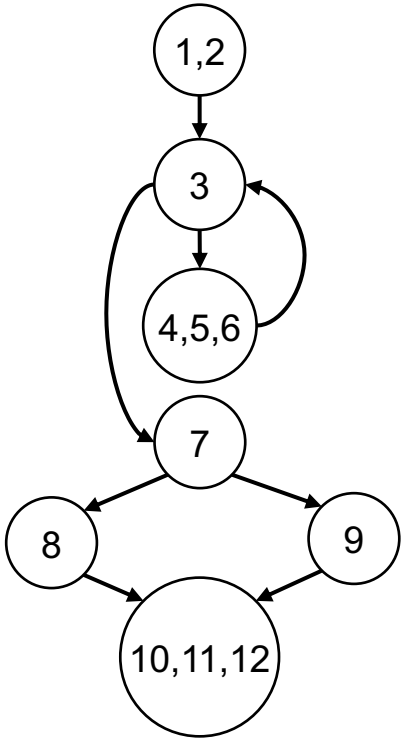
Control Flow Graph (CFG) *for a procedure*

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.        x = x + 2*y;
5.        call Third;
6.        i = i - 1; }
7.     if (x==1) then {
8.        x = x+1; }
       else {
9.        z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```



Use of Control Flow Graph (CFG)

```
procedure Second {
1.      x = 0;
2.      i = 5;
3.      while (i!=0)   {
4.         x = x + 2*y;
5.         call Third;
6.         i = i - 1; }
7.      if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```



- Is there an execution path from statement #2 to statement #9?
- If I change value of ‘i’ at statement #2 - which other statements will be affected?
- Which statements affect the value of ‘z’ at statement #12? directly? indirectly?

CFG for SIMPLE

- One CFG per procedure
- Take note of Statement vs. Program Lines
- Group statements in basic blocks if possible
  - While statement (head) should be in a separate node from other statements in the loop (body)
- Use dummy nodes (but not excessively)
  - Show exit node when it is not obvious
    - » If statements in the CFG have a diamond shape
    - » While loops have a loop shape

Next and Next*

Relationship Next defines the control flow:

For two program lines n1 and n2 in the same procedure

- **Next (n1, n2)** holds if n2 can be executed **immediately after** n1 in some execution sequence
- **Next* (n1, n2)** holds if n2 can be executed **after** n1 in some execution sequence

Examples of Next

Which relationships hold?

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.       x = x+1; }
        else {
9.       z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

A. Next (2, 3)

B. Next (3, 4)

C. Next (3, 7)

D. Next (5, 6)

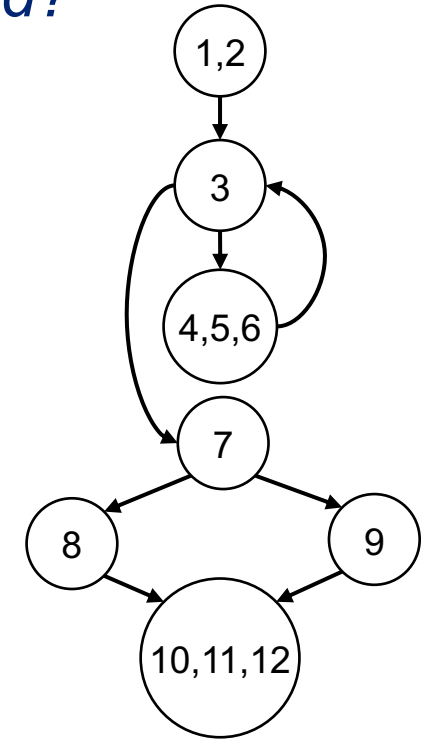
E. Next (6, 4)

F. Next (7, 9)

G. Next (7, 10)

H. Next (8, 9)

I. Next (8, 10)



Examples of Next

Which relationships hold?

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.       x = x+1; }
        else {
9.       z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

A. Next (2, 3). True

B. Next (3, 4) True

C. Next (3, 7) True

D. Next (5, 6) True

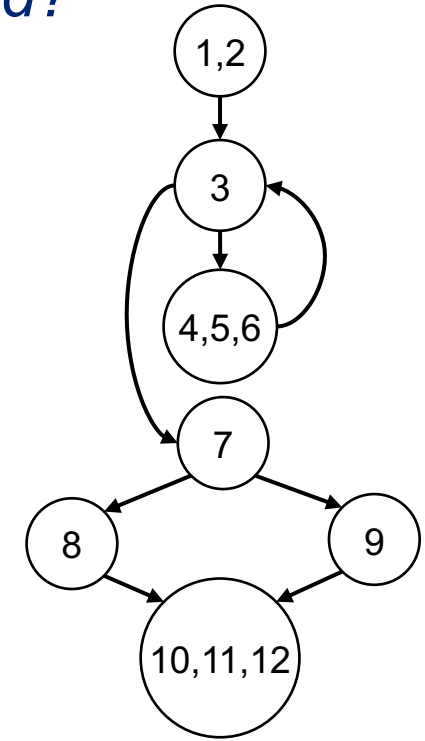
E. Next (6, 4) False

F. Next (7, 9) True

G. Next (7, 10) False

H. Next (8, 9) False

I. Next (8, 10) True



Examples of Next*

Which relationships hold?

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.       x = x+1; }
        else {
9.       z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

A. Next* (1, 2)

B. Next* (1, 3)

C. Next* (2, 5)

D. Next* (4, 3)

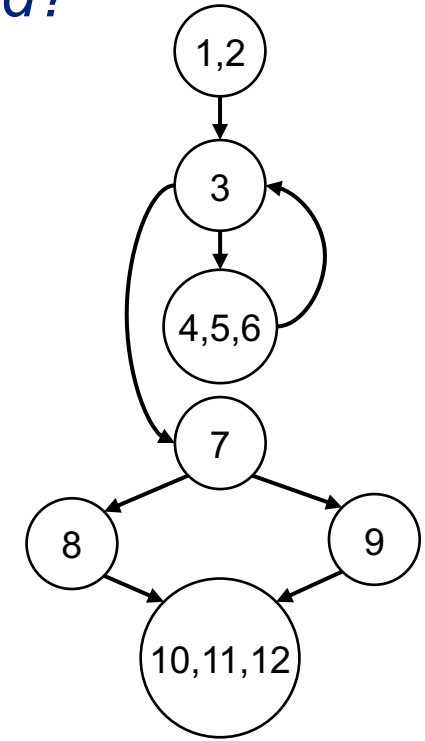
E. Next* (5, 2)

F. Next* (5, 5)

G. Next* (5, 8)

H. Next* (5, 12)

I. Next* (8, 9)



Examples of Next*

Which relationships hold?

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.       x = x+1; }
        else {
9.       z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

A. Next* (1, 2) True

B. Next* (1, 3) True

C. Next* (2, 5) True

D. Next* (4, 3) True

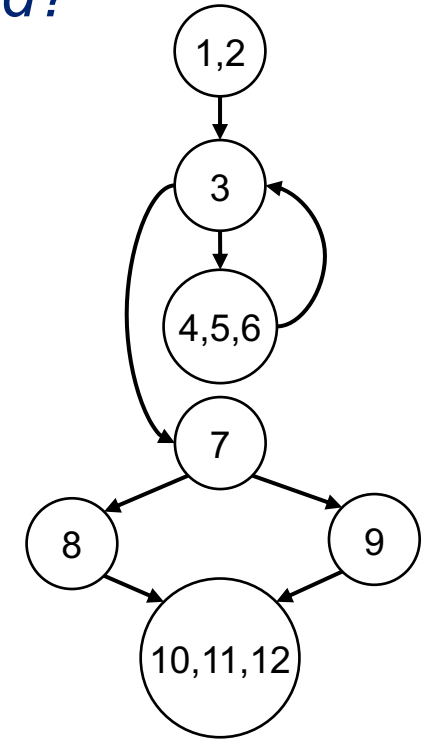
E. Next* (5, 2) False

F. Next* (5, 5) True

G. Next* (5, 8) True

H. Next* (5, 12) True

I. Next* (8, 9) False



Affects Definition

Affects (a1, a2) holds if:

- a1 and a2 are in the **same procedure**
- a1 modifies a variable **v** which is used in a2
- There is a **control flow path** from a1 to a2 on such that **v is not modified** (as defined by Modifies relationship) in any assignment, read, or procedure call statement on that path

Affects

Affects (a1, a2) holds if

- a1 modifies a variable v which is used in a2
- there is an execution path from a1 to a2 on which v is not modified

Affects (1, 2)?

```
1. x = 5;
2. y = x;
```

```
1. x = 5;
2. y = z;
```

Affects (2, 3)?

```
1. if (z>0) then{
2.   x = 5;}
   else {
3.   y = x;}
```

Affects (1, 3)?

```
1. x = 5;
2. x = 10;
3. y =  x;
```

Examples of Affects

Which relationships hold?

A. Affects (2, 3)

B. Affects (2, 6)

C. Affects (4, 8)

D. Affects (4, 10)

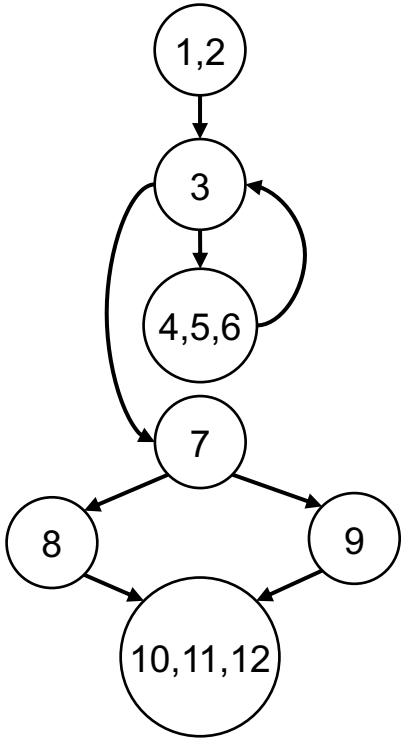
E. Affects (9, 6)

F. Affects (9, 11)

G. Affects (6, 6)

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
     z = 5;
     v = z;
     print v;   }
```



Examples of Affects

Which relationships hold?

A. Affects (2, 3) False

B. Affects (2, 6) True

C. Affects (4, 8) True

D. Affects (4, 10) True

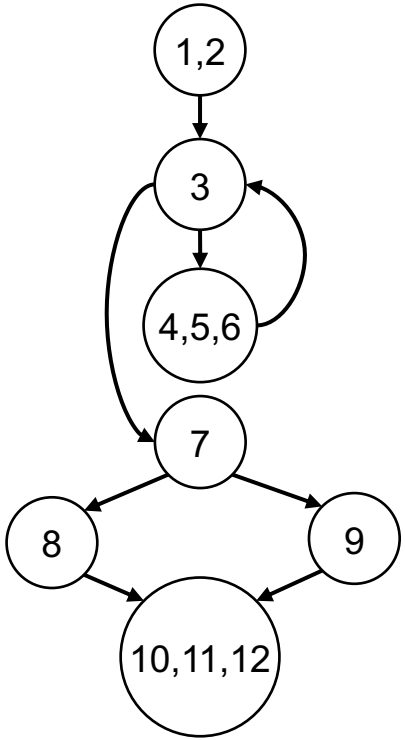
E. Affects (9, 6) False

F. Affects (9, 11) False

G. Affects (6, 6) True

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.       x = x + 2*y;
5.           call Third;
6.       i = i - 1; }
7.     if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
     z = 5;
     v = z;
     print v;   }
```



Even More Examples of Affects

```
procedure p {
1.  x = 1;
2.  y = 2;
3.  z = y;
4.  call q;
5.  z = x + y + z;}
```

```
procedure q {
6.  x = 5;
7.  t = 4;
8.  if (z>0) then
    {
9.  t = x + 1;}
    else {
10.   y = z + x;}
11. x = t + 1; }
```

Which relationships hold?

A. Affects (3, 5)

B. Affects (1, 5)

C. Affects (2, 5)

D. Affects (7, 11)

Affects* Definition

Affects* (a1, a2) holds if:

- Affects (a1, a2) or
- Affects (a1, a) and Affects* (a, a2) for some assignment statement a

Affects*

Affects* (a1, a2) holds if

- a1 affects a2 directly or indirectly

Example: Affects* (1, 2)

Affects* (2, 3)

Affects* (1, 3)

```
1.  x = a;
2.  v = x;
3.  z = v;
```

Examples of Affects*

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.         x = x + 2*y;
5.         call Third;
6.         i = i - 1; }
7.     if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
    z = 5;
    v = z;
    print v;  }
```

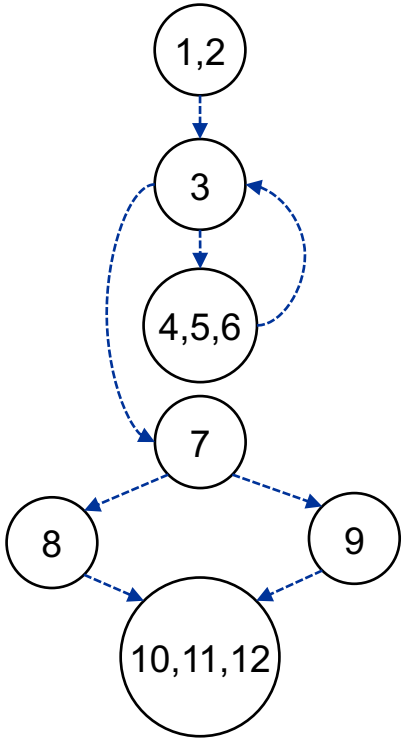
Which relationships hold?

A. Affects* (1, 4)

B. Affects* (1, 10)

C. Affects* (1, 11)

D. Affects* (9, 12)



Examples of Affects*

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.         x = x + 2*y;
5.         call Third;
6.         i = i - 1; }
7.     if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

```
procedure Third {
    z = 5;
    v = z;
    print v;  }
```

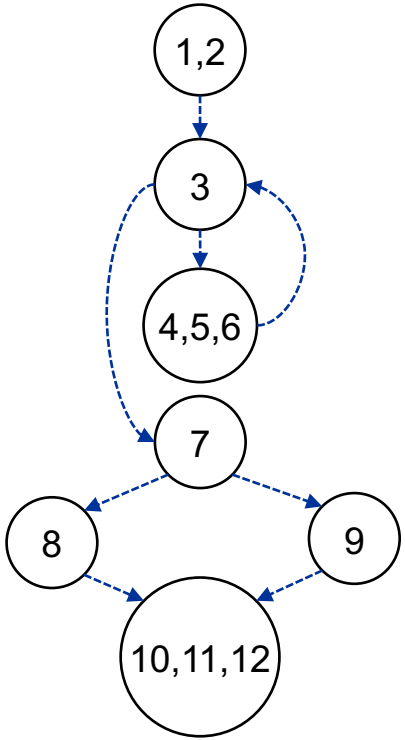
Which relationships hold?

A. Affects* (1, 4) True

B. Affects* (1, 10) True

C. Affects* (1, 11) True

D. Affects* (9, 12) True



New Information for PKB

Examples

Queries	What to store
Which procedures call a given procedure p? Directly or indirectly.	Store information about procedure-call relationships
Which statements have been executed before this statement?	Store information about CFG
Which assignments affect this variable value?	Relevant Information to store in the PKB

Program Design Abstractions

Design abstraction (Direct/Indirect relationship)	Relationship between
Follows/Follows*	Statements
Parent/Parent*	Statements
Calls/Calls*	Procedures
Next/Next*	Program lines
Affects/Affects*	Assignment statements

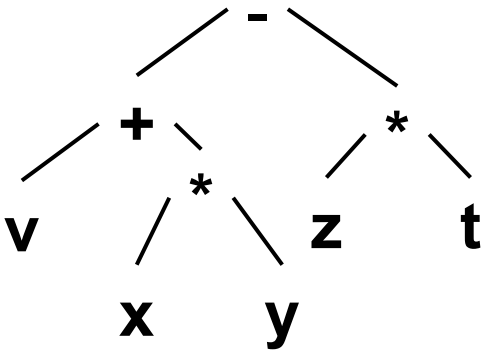
(recall) pattern

Which of the patterns match with this assignment statement?

x = v + x * y - z * t

A. a (_ , “v + x * y - z * t”)
B. a (_ , “v”)
C. a (_ , _“v”_)
D. a (_ , _“x * y”_)
E. a (_ , _“v + x”_)
F. a (_ , _“v + x *y”_)
G. a (_ , _“y - z *t”_)
H. a (_, _“x * y - z * t”_)
I. a (_ , _“v + x * y - z * t”_)

A. T
B. F
C. T
D. T
E. F
F. T
G. F
H. F
I. T



(AdvSPA) pattern

- Expressions
  - pattern a (_, _)
  - "_" stands for left-hand-side or right-hand-side of the assignment a
  - "_" can be replaced with references and/or expressions
  - check correctness using grammar of PQL
- Container statements – control variable v
  - If: pattern ifs (v, _, _)
  - While: pattern w (v, _)
  - "_" stands for statement list (is not defined)

Example

while w;

Select w pattern w(“i”, _)

Answer: 3

if ifs;

Select ifs pattern ifs(“x”, _, _)

Answer : 7

```
procedure Second {
1.     x = 0;
2.     i = 5;
3.     while (i!=0)  {
4.         x = x + 2*y;
5.         call Third;
6.         i = i - 1; }
7.     if (x==1) then {
8.         x = x+1; }
        else {
9.         z = 1; }
10.   z = z + x + i;
11.   y = z + 2;
12.   x = x * y + z; }
```

More on PQL side

Queries in PQL contain:

- Declaration
  - more types of synonyms
- Select clause specifies query result
  - single or multiple return values (tuples) or BOOLEAN
  - such that clauses constrains the results in terms of relationships
  - pattern clauses contains results in terms of code patterns
  - with clause constrains the results in terms of attribute values
- Query results must check (make true) all “such that”, “with” and “pattern” clauses

PQL grammar – some notable changes

- attrName : ‘procName’| ‘varName’ | ‘value’ | ‘stmt#’
- design-entity : ‘stmt’ | ‘read’ | ‘print’ | ‘call’ | ‘while’ | ‘if’ | ‘assign’ | ‘variable’ | ‘constant’ | ‘prog_line’ | ‘procedure’
- select-cl : declaration* ‘Select’ result-cl (suchthat-cl | with-cl | pattern-cl)*
- result-cl : tuple | ‘BOOLEAN’
- tuple: elem | ‘<’ elem (‘,’ elem)* ‘>’
- with-cl : ‘with’ attrCond
- attrCond : attrCompare (‘and’ attrCompare)*
- attrCompare : ref ‘=’ ref

// the two refs must be of the same type (i.e., both strings or both integers)

- ref : ‘”’ IDENT ‘”’ | INTEGER | attrRef | synonym

// synonym must be of type ‘prog_line’

- relRef : ModifiesP | ModifiesS | UsesP | UsesS | Calls | CallsT | Parent | ParentT | Follows | FollowsT | Next | NextT | Affects | AffectsT
- Next : ‘Next’ ‘(’ lineRef ‘,’ lineRef ‘)’
- NextT : ‘Next*’ ‘(’ lineRef ‘,’ lineRef ‘)’
- Affects : ‘Affects’ ‘(’ stmtRef ‘,’ stmtRef ‘)’
- AffectsT : ‘Affects*’ ‘(’ stmtRef ‘,’ stmtRef ‘)’
- patternCond : pattern (‘and’ pattern)*
- pattern : assign | while | if
- if : syn-if ‘(’ entRef ‘,’ ‘_’ ‘,’ ‘_’ ‘)’
- // syn-if must be of type ‘if’
- while : syn-while ‘(’ entRef ‘,’ ‘_’ ‘)’
- // syn-while must be of type ‘while’

PQL grammar – some notable changes

- Query result can be single or multiple values (tuples) or BOOLEAN
- Clauses “such that”, “with” and “pattern” may occur many times in the same query.
  - Select <….>
  - Select <…..> such that <……> pattern <……> with <…..>
  - Select <…..> pattern <….> such that <……> pattern <……> with <…..>
  - Select <…..> pattern <….> pattern <…> with <…..> such that <……> such that <….>
- There is an implicit **and** operator between different types of clauses – that means a query result must satisfy the conditions specified in all the clauses.
  - Select <…..> such that <……> pattern <……> with <…..>

PQL grammar – some notable changes

- Operator and can be explicitly used
  - in place of ‘such that’ to combine two ‘such that’ clauses
    - » Select <…..> such that <……> such that <……>
    - » Select <…..> such that <……> and <……>
  - in place of ‘with’ to combine two ‘with’ clauses
    - » Select <…..> with <……> with <……>
    - » Select <…..> with <……> and <……>
  - In place of ‘pattern’ to combine two ‘pattern’ clauses
    - » Select <…..> pattern <……> pattern <……>
    - » Select <…..> pattern <……> and <……>

Example - procedure synonym, Calls/Calls*

Q: What are the procedures in the program call another procedure?

procedure p, q;

Select p such that Calls (p, _) or Select p such that Calls (p, q)

Answer: First, Second

Q: Which procedures call procedure “Third” directly or indirectly?

procedure p;

Select p such that Calls (p, “Third”)*

Answer: First, Second

Example - Affects/Affects*

Which assignments directly or indirectly affect value computed at assignment #10?

assign a;

Select a such that Affects (a, 10)*

- Answer: 9, 1, 4, 8, 2, 6

Example – prog_line synonym, Next/Next*, multiple such that clauses

Q: Which program lines can be executed between line #5 and line #12?

prog_line n;

Select n such that Next (5, n) and Next* (n, 12)*

Answer: 3,4,5,6,7,8,9,10,11

** *Select n such that Next* (5, n) such that Next* (n, 12)*

Another Example – multi clause

Q: Find assignments to variable “x” located in a loop, that can be reached (in terms of control flow) from statement #1.

assign a; while w;

Select a pattern a (“x”, _) such that Parent (w, a) and Next* (1, a)*

Answer: 4

Alternatively :

Select a such that Modifies (a, “x”) and Parent (w, a) and Next* (1, a)*

Select a pattern a (“x”, _) such that Parent (w, a) such that Next* (1, a)*

Select a such that Parent (w, a) and Next* (1, a) pattern a (“x”, _)*

Example- with clause

Q: Which procedures are called from “Second” in a while loop?

procedure p; call c; while w;

Select p such that Calls (“Second”, p) and Parent(w, c) with c.procName = p. procName

- Answer: Third

** Take note of attribute procName

attrName : ‘procName’| ‘varName’ | ‘value’ | ‘stmt#’

Example- with clause

Q: Which procedures directly call procedure Third and modify i?

procedure p, q;

Select p such that Calls (p, q) with q.procName = “Third” such that Modifies (p, “i”)

- Answer: Second

** Take note of attribute procName

attrName : ‘procName’| ‘varName’ | ‘value’ | ‘stmt#’

Examples – Boolean

Q: Is there an execution path from statement #2 to statement #9?

Select BOOLEAN such that Next (2, 9)*

- Answer: TRUE

Q: Is there an execution path from statement #2 to statement #9 that passes through statement #8?

Select BOOLEAN such that Next (2, 8) and Next* (8, 9)*

- Answer: FALSE

Examples – Tuples

//Select <a, p, s> …… returns a list of strings. (refer to Auto-tester description in the wiki)

Q: Find all pairs of procedures p and q such that p calls q.

procedure p, q;

Select <p, q> such that Calls (p, q)

Answer: *First Second, Second Third*

Result tuples are separated by a comma.
Elements within each tuple are separated by a space.
There is a space between the comma and the next tuple.

Examples – Tuples

//Following *Query returns all the instances of three nested while loops in a program, not just the first one that is found.*

while w1, w2, w3;

Select <w1, w2, w3> such that Parent* (w1, w2) and Parent* (w2, w3)

//Following query returns all pairs of assignments that affect each other?

assign a1, a2;

Select <a1, a2> such that Affects (a1, a2)

or

Select <a1.stmt#, a2> such that Affects (a1, a2)

Comment on Program Queries

- Changing the order of conditions in a query does not change the query result

assign a; while w;

Select a such that Modifies (a, “x”) and Parent* (w, a) and Next* (1, a)

Select a such that Parent* (w, a) and Modifies (a, “x”) such that Next* (1, a)

Select a such that Next* (1, a) and Parent* (w, a) and Modifies (a, “x”)

Select a pattern a (“x”, _) such that Parent* (w, a) such that Next* (1, a)

Select a such that Parent* (w, a) and Next* (1, a) pattern a (“x”, _)

Select a such that Next* (1, a) and Parent* (w, a) pattern a (“x”, _)

- Answer: 4

BUT: Changing the order of conditions may affect query evaluation time

Format of Result

Select	Should return
BOOLEAN	TRUE/FALSE
Statement s(a/n/c/w/ifs) Program line n	Statement number
Variable v Procedure p	Names (no need to use “”)
Constant ct	Constant value
StmtLst sl	Number of the first statement in the statement list
Tuple <s, v>	Tuple values (e.g. “2 x, 3 z,…”, etc)

Empty Result vs. Invalid Queries

- On paper/tests:
  - Empty result can be represented using **None**, Nil, Null keywords
  - Invalid queries should be identified
- In AutoTester (SPA implementation):
  - Empty result is returned by not populating the list of strings in the TestWrapper (evaluate function) – return an empty list of stings
  - Invalid queries should also return an empty result

Summary – Advanced SPA

- More relationships

Design Abstraction	Relationship between
Calls/Calls*	Procedures
Next/Next*	Program lines
Affects/Affects*	Assignments

- pattern for container- statements
- More on PQL side
  - Supports the additional relationships and patterns
  - Multi-clause – such that, pattern, with
  - Query results : single, multiple, BOOLEAN