Solution to Homework 3

Does criterion C_1 subsume C_2? That is, if a test set gives 100% coverage with respect to C_1, will it also give 100% coverage with respect to C_2?

	block	statement	condition	decision	CD	MC	c-use	p-use	all-use
block		Yes (1)	No (2)	Yes (3)	Yes (4)	Yes (5)	No (6)	Yes (7)	Yes (8)
statement	Yes (1)		No (2)	Yes (3)	Yes (4)	Yes (5)	No (6)	Yes (7)	Yes (8)
condition	No (2)	No (2)		No (9)	Yes (10)	Yes (11)	No (12)	No (13)	No (14)
decision	No (3)	No (3)	No (9)		Yes (15)	Yes (16)	No (17)	Yes (18)	Yes(19)
CD	No (4)	No (4)	No (10)	No (15)		Yes (20)	No (21)	No (22)	No (23)
MC	No (5)	No (5)	No (11)	No (16)	No (20)		No (24)	No (25)	No (26)
c-use	No (6)	No (6)	No (12)	No (17)	No (21)	No (24)		No (27)	Yes(28)
p-use	No (7)	No (7)	No (13)	No (18)	No (22)	No (25)	No (27)		Yes(29)
all-use	No (8)	No (8)	No (14)	No (19)	No (23)	No (26)	No (28)	No (29)	

1. block and statement

- 100% block coverage gives 100% statement coverage? → **Yes**

  Use contradiction method.

  ➔ Make an assumption: a test suite gives 100% block coverage but not 100% statement coverage
  ➔ This implies that every block is covered but some statements are still not covered
  ➔ Some statements do not belong to any blocks
  ➔ A contradiction is found ➔ our assumption is not correct
  ➔ a test suite that gives 100% block coverage must give 100% statement coverage

- 100% statement coverage gives 100% block coverage? → **Yes**

  Use the similar contradiction method shown above.

2. block/statement and condition

condition coverage = all the possibilities of every condition (i.e., T & F) in every decision needs to be covered

- 100% block/statement coverage gives 100% condition coverage? → **No**

```
...
if (a && b)
{
...
}
return 0;
```

Any test case that makes the if decision true can get 100% statement coverage but not 100% condition coverage.

- 100% condition coverage gives 100% block/statement coverage? → **No**

```
...
if (a && b)
{
...
}
return 0;
```

** conditions **	** statement **
t_1: T F t_2: F T all possibilities of each condition are covered ➔ 100% condition coverage	statement in the true branch was never executed ➔ Not 100% block/statement coverage.

3. block/statement and decision

   - 100% block/statement coverage gives 100% decision coverage? → **No**

     A simple counter example: (a = T, b = T)

   - 100% decision coverage gives 100% block/statement coverage? → **Yes**

     Every block/statement must be part of a branch; or it will be covered by any test cases. So covering both true and false branches of all decisions makes every block/statement to be covered.

4. block/Statement and condition-decision

   - 100% block/statement coverage gives 100% condition-decision coverage? → **No**

     Since 100% block/statement coverage cannot guarantee 100% condition or decision coverage, it cannot give 100% condition-decision coverage (which subsumes condition coverage and decision coverage).

   - 100% condition-decision coverage gives 100% block/statement coverage? → **Yes**

     Since 100% decision coverage gives 100% block/statement coverage, 100% condition-decision (which subsumes condition coverage and decision coverage) also gives 100% block/statement coverage.

5. block/Statement and multiple-condition

   - 100% block/statement coverage gives 100% multiple-condition coverage? → **No**

```
s1:   if (a > 0 && b > 0){
s2:     a++;
s3:   }else{
s4:     b++;
s5:   }
```

     100% block cannot guarantee 100% condition
     ➔ cannot give 100% multiple-condition coverage which is a stronger version of condition coverage

   - 100% multiple-condition coverage gives 100% block/statement coverage? → **Yes**

Multiple-condition coverage requires every combination of simple condition to be tried once.
➔ two outcomes of every decision to be covered ➔ 100% decision
➔ 100% block-statement coverage

6. block/statement and c-use

- 100% block/statement coverage gives 100% c-use coverage? → **No**

```
s1:    if (a > 0) {
s2:      x = 1
s3:    }else{
s4:      x = 2;
s5:    }
s6:    if (b > 0) {
s7:      y = x + 1;
s8:    }else{
s9:      y = x – 1;
s10:   }
```

The define/c-use pair of variable x contains (s_2, s_7), (s_2, s_9), (s_4, s_7), (s_4, s_9).
t_1: {a = -1, b = 1} and t_2: {a =1, b = -1} gives 100% block/statement coverage. However, only two pairs are covered (s_2, s_9), (s_4, s_7).

- 100% c-use coverage gives 100% block/statement coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0){
s3:      print (a + b);
s4:    }else{
s5:      print ("Hello World!");
s6:    }
```

s_5 does not contain c-use of any variable. t_1: {a = 1, b = 1} gives 100% c-use coverage. However, s_5 is not covered.

7. block/statement and p-use

- 100% block/statement coverage gives 100% p-use coverage? → **No**

```
s1:    input (x);
s2:    if (x > 0) {
s3:      y = x + 1;
s4:    }
s5:    print(y);
```

t_1: {x = 1} gives 100% block/statement coverage. 100% p-use coverage requires define/p-use pairs (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)) to be covered. However, only one of them, (s_1, (s_2, s_3)) is covered.

- 100% p-use coverage gives 100% block/statement coverage? → **Yes**

```
s1:   input (x, y);
s2:   if (x > 0) {
s3:     x++;
s4:   }
s5:   print(x);
s6:   if (y > 0) {
s7:     y++;
s8:   }
s9:   print(y);
```

100% p-use coverage guarantees that define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable x are covered and define/p-use pairs, (s_1, (s_6, s_7)) and (s_1, (s_6, s_9)), of variable y are covered. It implies that all program branches are covered. Hence, all block/statements are also covered.

8. block/statement and all-use

- 100% block/statement coverage gives 100% all-use coverage? → **No**

  Since 100% block/statement coverage does not give 100% c-use coverage or 100% p-use coverage, it cannot give 100% all-use coverage (which subsumes c-use and p-use coverage).

- 100% all-use coverage gives 100% block/statement coverage? → **Yes**

  Since 100% p-use coverage give 100% block/statement coverage, 100% all-use coverage also gives 100% block/statement coverage.

9. condition and decision

- 100% condition coverage gives 100% decision coverage? → **No**

  A simple counter example:

  if (a && b)

** conditions **	** decision **
t_1: T F t_2: F T all possibilities of each condition are covered ➔ 100% condition coverage	only the false branch is covered ➔ Not 100% decision coverage.

- 100% decision coverage gives 100% condition coverage? → **No**

  A simple counter example:

  if (a && b)

** decision **	** condition **
t_1: T F ➔ T t_2: F F ➔ F ➔ 100% decision coverage	T outcome of the condition "b" is not covered. ➔ Not 100% condition coverage

10. condition and condition-decision

- 100% condition coverage gives 100% condition-decision coverage? → **No**

  Since 100% condition cannot give 100% decision coverage, it also cannot give 100% condition-decision coverage (which subsumes condition and decision coverage).

- 100% condition-decision coverage gives 100% condition coverage? → **Yes**

  Condition-decision coverage subsumes condition coverage and decision coverage.

11. condition and multiple-condition

- 100% condition coverage gives 100% multiple-condition coverage? → **No**

  Condition coverage cares about whether all the outcomes of each condition are EVENTUALLY covered, but it does not care HOW it is covered.

  Multiple-condition coverage cares about not only "whether all the outcomes ... EVENTUALLY covered" but also "HOW ... are covered" ← the definition specifically indicates every combination should be covered at least once.

  A counter example can be easily found.

- 100% multiple-condition coverage gives 100% condition coverage? → **Yes**

  If multiple-condition cares about EVENTUALLY and HOW, then EVENTUALLY is guaranteed.
  → 100% condition coverage

12. condition and c-use

- 100% condition coverage gives 100% c-use coverage? → **No**

```
s1:   input (a, b); x = 0; y = 0;
s2:   if (a > 0 || b > 0){
s3:     a = x + 1;
s4:   }else{
s5:     b = y + 1;
s6:   }
```

  The define/c-use pair of variable x contains (s_1, s_3).
  The define/c-use pair of variable y contains (s_1, s_5).
  t_1: {a = -1, b = 1} and t_2: {a =1, b = -1} gives 100% condition coverage. However, the define/c-use pair of y is not covered.

- 100% c-use coverage gives 100% condition coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0){
s3:      print (a + b);
s4:    }else{
s5:      print ("Hello World!");
s6:    }
```

s_5 does not contain c-use of any variable. t_1: {a = 1, b = 1} gives 100% c-use coverage. However, t_1 cannot make $a > 0$ as false.

13. condition and p-use

- 100% condition coverage gives 100% p-use coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0 || b > 0){
s3:      x ++;
s4:    }else{
s5:      y ++;
s6:    }
```

100% p-use coverage requires define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable a and define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable b to be covered.
t_1: {a = -1, b = 1}, t_2: {a =1, b = -1} gives 100% condition coverage. However, only (s_1, (s_2, s_3)) of a and (s_1, (s_2, s_3)) of b are covered.

- 100% p-use coverage gives 100% condition coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0 || b > 0){
s3:      x ++;
s4:    }else{
s5:      y ++;
s6:    }
```

100% p-use coverage requires define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable a and define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable b to be covered.
t_1: {a = -1, b = 1}, t_2: {a =-1, b = -1} gives 100% p-use coverage. However, $a > 0$ cannot be true using these test cases.

14. condition and all-use

- 100% condition coverage gives 100% all-use coverage? → **No**

Since 100% condition cannot give 100% p-use or 100% c-use coverage, it also cannot give 100% all-use coverage (which subsumes c-use and p-use coverage).

- 100% all-use coverage gives 100% condition coverage? → **No**

s_1: input (a, b, x, y);
s_2: if ($a > 0 \parallel b > 0$){
s_3: x ++;
s_4: }else{
s_5: y ++;
s_6: }

100% p-use coverage requires define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable a and define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable b to be covered.

The define/c-use pair of variable x contains (s_1, s_3). The define/c-use pair of variable y contains (s_1, s_5).

t_1: {a = -1, b = 1}, t_2: {a =-1, b = -1} gives 100% all-use (100% c-use and 100% p-use) coverage. However, $a > 0$ cannot be true using these test cases.

15. decision and condition-decision

- 100% decision coverage gives 100% condition-decision coverage? → **No**

  Since 100% decision cannot give 100% condition coverage, it also cannot give 100% condition-decision coverage (which subsumes condition and decision coverage).

- 100% condition-decision coverage gives 100% condition coverage? → **Yes**

  Condition-decision coverage subsumes condition coverage and decision coverage.

16. decision and multiple-condition

- 100% decision coverage gives 100% multiple-condition coverage? → **No**

  s_1: if ($a > 0 \parallel b > 0$){
  s_2: a++;
  s_3: }else{
  s_4: b++;
  s_5: }

  100% Multiple-condition coverage requires test cases which cover {$a > 0$ as true, $b > 0$ as true}, {$a > 0$ as true, $b > 0$ as false}, {$a > 0$ as false, $b > 0$ as true}, and {$a > 0$ as false, $b > 0$ as false}.
  t_1: {a = -1, b = 1}, t_2: {a = -1, b = -1} gives 100% decision coverage. However, they cannot cover {$a > 0$ as true, $b > 0$ as true} and {$a > 0$ as true, $b > 0$ as false}.

- 100% multiple-condition coverage gives 100% decision coverage? → **Yes**

  s_1: if ($a > 0 \parallel b > 0$){
  s_2: a++;
  s_3: }else{
  s_4: b++;
  s_5: }

100% Multiple-condition coverage requires test cases which cover {$a > 0$ as true, $b > 0$ as true}, {$a > 0$ as true, $b > 0$ as false}, {$a > 0$ as false, $b > 0$ as true}, and {$a > 0$ as false, $b > 0$ as false}. If 100% multiple-condition coverage is satisfied then all decision must be evaluated as both true and false.

17. decision and c-use

- 100% decision coverage gives 100% c-use coverage? → **No**

```
s1:    if (a > 0) {
s2:      x = 1
s3:    }else{
s4:      x = 2;
s5:    }
s6:    if (b > 0) {
s7:      y = x + 1;
s8:    }else{
s9:      y = x – 1;
s10:   }
```

The define/c-use pair of variable x contains (s_2, s_7), (s_2, s_9), (s_4, s_7), (s_4, s_9).
t_1: {a = -1, b = 1}, t_2: {a =1, b = -1} gives 100% decision coverage. However, only two pairs are covered (s_2, s_9), (s_4, s_7).

- 100% c-use coverage gives 100% decision coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0){
s3:      print (a + b);
s4:    }else{
s5:      print ("Hello World!");
s6:    }
```

s_5 does not contain c-use of any variable. t_1: {a = 1, b = 1} gives 100% c-use coverage. However, t_1 cannot make $a > 0$ as false.

18. decision and p-use

- 100% decision coverage gives 100% p-use coverage? → **No**

```
s1:    input (a);
s2:    if (a > 2) {
s3:      x = a * 2;
s4:    }else{
s5:      x = a + 6;
s6:    }
s7:    if (x > 7) {
s8:      y = x + 1;
s9:    }else{
s10:     y = x – 1;
s11:   }
```

100% p-use coverage requires define/p-use pairs, (s_3, (s_7, s_8)), (s_3, (s_7, s_{10})), (s_5, (s_7, s_8)), and (s_5, (s_7, s_{10})) of variable x are covered.
t_1: {$a = 4$}, t_2: {$a = 1$} gives 100% decision coverage.
t_1 covers (s_3, (s_7, s_8)) of x.
t_2 covers (s_5, (s_7, s_{10})) of x.
(s_3, (s_7, s_{10})) and (s_5, (s_7, s_8)) are not covered.

- 100% p-use coverage gives 100% decision coverage? → **Yes**

```
s1:     input (a);
s2:     if (a > 2) {
s3:       x = a * 2;
s4:     }else{
s5:       x = a + 6;
s6:     }
s7:     if (x > 7) {
s8:       y = x + 1;
s9:     }else{
s10:      y = x – 1;
s11:    }
```

100% p-use coverage requires define/p-use pairs, (s_3, (s_7, s_8)), (s_3, (s_7, s_{10})), (s_5, (s_7, s_8)), and (s_5, (s_7, s_{10})) of variable x are covered.
t_1: {$a = 4$}, t_2: {$a =1$}, t_3: {$a = 3$}, t_4: {$a =2$} gives 100% p-use coverage. They also guarantee 100% decision coverage.

Generally speaking, for p-use coverage, every p-use must independently contribute to the decision taking both a true and false branch. So it guarantees decision coverage.

19. decision and all-use

- 100% decision coverage gives 100% all-use coverage? → **No**

  Since 100% decision coverage cannot give 100% p-use or 100% c-use coverage, it also cannot give 100% all-use coverage (which subsumes c-use and p-use coverage).

- 100% all-use coverage gives 100% decision coverage? → **Yes**

  Since 100% p-use coverage gives 100% decision coverage, 100% all-use coverage (which subsumes p-use coverage) also gives 100% decision coverage.

20. condition-decision and multiple-condition

- 100% condition-decision coverage gives 100% multiple-condition coverage? → **No**

```
s1:     if (a > 0 || b > 0){
s2:       a++;
s3:     }else{
s4:       b++;
```

s_5: }

100% Multiple-condition coverage requires test cases which cover {$a > 0$ as true, $b > 0$ as true}, {$a > 0$ as true, $b > 0$ as false}, {$a > 0$ as false, $b > 0$ as true}, and {$a > 0$ as false, $b > 0$ as false}.
t_1: {$a = -1$, $b = 1$}, t_2: {$a = -1$, $b = -1$}, t_3: {$a = 1$, $b = -1$} gives 100% condition-decision coverage. However, they cannot cover {$a > 0$ as true, $b > 0$ as true}.

- 100% multiple-condition coverage gives 100% condition-decision coverage? → **Yes**

```
s1:   if (a > 0 || b > 0){
s2:     a++;
s3:   }else{
s4:     b++;
s5:   }
```

100% Multiple-condition coverage requires test cases which cover {$a > 0$ as true, $b > 0$ as true}, {$a > 0$ as true, $b > 0$ as false}, {$a > 0$ as false, $b > 0$ as true}, and {$a > 0$ as false, $b > 0$ as false}.
If 100% multiple-condition coverage is satisfied, it implies all possible combinations of condition outcomes are covered so that 100% condition-decision coverage is achieved.

21. condition-decision and c-use

- 100% condition-decision coverage gives 100% c-use coverage? → **No**

```
s1:   if (a > 0) {
s2:     x = 1
s3:   }else{
s4:     x = 2;
s5:   }
s6:   if (b > 0) {
s7:     y = x + 1;
s8:   }else{
s9:     y = x – 1;
s10:  }
```

The define/c-use pair of variable x contains (s_2, s_7), (s_2, s_9), (s_4, s_7), (s_4, s_9).
t_1: {$a = -1$, $b = 1$}, t_2: {$a = 1$, $b = -1$} gives 100% condition-decision coverage. However, only two pairs are covered (s_2, s_9), (s_4, s_7).

- 100% c-use coverage gives 100% condition-decision coverage? → **No**

```
s1:   input (a, b);
s2:   if (a > 0){
s3:     print (a + b);
s4:   }else{
s5:     print ("Hello World!");
s6:   }
```

s_5 does not contain c-use of any variable. t_1: {$a = 1$, $b = 1$} gives 100% c-use coverage. However, t_1 cannot make $a > 0$ as false.

22. condition-decision and p-use

- 100% condition-decision coverage gives 100% p-use coverage? → **No**

```
s1:    input (a);
s2:    if (a > 2) {
s3:      x = a * 2;
s4:    }else{
s5:      x = a + 6;
s6:    }
s7:    if (x > 7) {
s8:      y = x + 1;
s9:    }else{
s10:     y = x – 1;
s11:   }
```

100% p-use coverage requires define/p-use pairs, (s_3, (s_7, s_8)), (s_3, (s_7, s_{10})), (s_5, (s_7, s_8)), and (s_5, (s_7, s_{10})) of variable x are covered.
t_1: {a = 4}, t_2: {a = 1} gives 100% condition-decision coverage.
t_1 covers (s_3, (s_7, s_8)) of x.
t_2 covers (s_5, (s_7, s_{10})) of x.
(s_3, (s_7, s_{10})) and (s_5, (s_7, s_8)) are not covered.

- 100% p-use coverage gives 100% condition-decision coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0 || b > 0){
s3:      x ++;
s4:    }else{
s5:      y ++;
s6:    }
```

100% p-use coverage requires define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable a and define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable b are covered.
t_1: {a = -1, b = 1}, t_2: {a =-1, b = -1} gives 100% p-use coverage. However, $a > 0$ cannot be true using these test cases. So 100% condition-decision coverage is not satisfied.

23. condition-decision and all-use

- 100% condition-decision coverage gives 100% all-use coverage? → **No**

  Since 100% condition-decision coverage cannot give 100% p-use or 100% c-use coverage, it also cannot give 100% all-use coverage (which subsumes c-use and p-use coverage).

- 100% all-use coverage gives 100% condition-decision coverage? → **No**

  Since 100% all-use coverage cannot give 100% condition coverage, Hence 100% all-use coverage also cannot give 100% condition-decision coverage (which subsumes decision and condition coverage).

24. multiple-condition and c-use

- 100% multiple-condition coverage gives 100% c-use coverage? → **No**

```
s1:    if (a > 0) {
s2:      x = 1
s3:    }else{
s4:      x = 2;
s5:    }
s6:    if (b > 0) {
s7:      y = x + 1;
s8:    }else{
s9:      y = x – 1;
s10:   }
```

The define/c-use pair of variable x contains (s_2, s_7), (s_2, s_9), (s_4, s_7), (s_4, s_9).
t_1: {a = -1, b = 1}, t_2: {a =1, b = -1} gives 100% multiple-condition coverage. However, only two pairs are covered (s_2, s_9), (s_4, s_7).

- 100% c-use coverage gives 100% multiple-condition coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0){
s3:      print (a + b);
s4:    }else{
s5:      print ("Hello World!");
s6:    }
```

s_5 does not contain c-use of any variable. t_1: {a = 1, b = 1} gives 100% c-use coverage. However, t_1 cannot make a > 0 as false.

25. multiple-condition and p-use

- 100% multiple-condition coverage gives 100% p-use coverage? → **No**

```
s1:    input (a);
s2:    if (a > 2) {
s3:      x = a * 2;
s4:    }else{
s5:      x = a + 6;
s6:    }
s7:    if (x > 7) {
s8:      y = x + 1;
s9:    }else{
s10:     y = x – 1;
s11:   }
```

100% p-use coverage requires define/p-use pairs, (s_3, (s_7, s_8)), (s_3, (s_7, s_{10})), (s_5, (s_7, s_8)), and (s_5, (s_7, s_{10})) of variable x are covered.
t_1: {a = 4}, t_2: {a = 1} gives 100% multiple-condition coverage.

t_1 covers (s_3, (s_7, s_8)) of x.
t_2 covers (s_5, (s_7, s_{10})) of x.
(s_3, (s_7, s_{10})) and (s_5, (s_7, s_8)) are not covered.

- 100% p-use coverage gives 100% multiple-condition coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0 || b > 0){
s3:      x ++;
s4:    }else{
s5:      y ++;
s6:    }
```

100% p-use coverage requires define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable a and define/p-use pairs, (s_1, (s_2, s_3)) and (s_1, (s_2, s_5)), of variable b are covered.
t_1: {a = -1, b = 1}, t_2: {a =-1, b = -1} gives 100% p-use coverage. However, to satisfy 100% multiple condition coverage, test cases need to cover {a > 0 as true, b > 0 as true}, {a > 0 as true, b > 0 as false}, {a > 0 as false, b > 0 as true}, and {a > 0 as false, b > 0 as false}. Two test cases obviously cannot achieve such coverage.

26. multiple-condition and all-use

- 100% multiple-condition coverage gives 100% all-use coverage? → **No**

Since 100% multiple-condition coverage cannot give 100% p-use or 100% c-use coverage, it also cannot give 100% all-use coverage (which subsumes c-use and p-use coverage).

- 100% all-use coverage gives 100% multiple-condition coverage? → **No**

Since 100% all-use coverage cannot give 100% condition coverage, Hence 100% all-use coverage also cannot give 100% multiple-condition coverage.

27. c-use and p-use

- 100% c-use coverage gives 100% p-use coverage? → **No**

```
s1:    input (a, b);
s2:    if (a > 0){
s3:      print (a + b);
s4:    }else{
s5:      print ("Hello World!");
s6:    }
```

s_5 does not contain c-use of any variable. t_1: {a = 1, b = 1} gives 100% c-use coverage. However, define/p-use pair of variable a, (s_1, (s_2, s_5)), is not covered.

- 100% p-use coverage gives 100% c-use coverage? → **No**

```
s1:    input (a, b)
s2:    if (a > 0) {
```

s_3: $x = 1$
s_4: }else{
s_5: $x = 2$;
s_6: }
s_7: if ($b > 0$) {
s_8: $y = x + 1$;
s_9: }else{
s_{10}: $y = x - 1$;
s_{11}: }

The define/c-use pair of variable x contains (s_3, s_8), (s_3, s_{10}), (s_5, s_8), (s_5, s_{10}).
The define/p-use pair of variable a contains (s_1, (s_2, s_3)), (s_1, (s_2, s_5))
The define/p-use pair of variable b contains (s_1, (s_7, s_8)), (s_1, (s_7, s_{10}))
t_1: {a = -1, b = 1}, t_2: {a =1, b = -1} gives 100% p-use coverage. However, two define/c-use pairs of x are not covered (s_3, s_8), (s_5, s_{10}).

28. c-use and all-use

- 100% c-use coverage gives 100% all-use coverage? → **No**

  100% c-use does not give 100% p-use (which is subsumed by all-use).

- 100% all-use coverage gives 100% c-use coverage? → **Yes**

  All-use subsumes c-use.

29. p-use and all-use

- 100% c-use coverage gives 100% all-use coverage? → **No**

  100% p-use does not give 100% c-use (which is subsumed by all-use).

- 100% all-use coverage gives 100% c-use coverage? → **Yes**

  All-use subsumes p-use.