

# Softwarequalität

## **Vorlesung 12 – Testen: White-box-Verfahren (Datenflussorientierte Verfahren)**

*Prof. Dr. Joel Greenyer*



27. Juni 2016



# Testüberdeckung (Coverage)

In der letzten Vorlesung...

- **Ziel:** Kenntnis der internen Struktur systematisch nutzen



# Testüberdeckung (Coverage)

In der letzten Vorlesung...

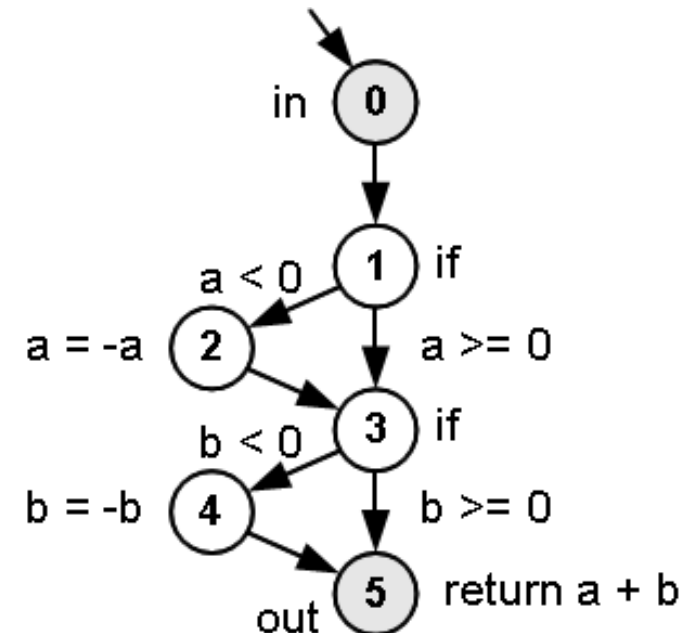
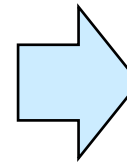
- **Ziel:** Kenntnis der internen Struktur systematisch nutzen
- **Kontrollflussorientierte Tests:**
  - Wie viele **Anweisungen** des Programms wurden abgedeckt?
  - Wie viele **Zweige** des Programms wurden abgedeckt?
  - Wie viele **Pfade** des Programms wurden abgedeckt?
  - Wie viele **Bedingungen** des Programms wurden abgedeckt?
  - ...

- **Ziel:** Kenntnis der internen Struktur systematisch nutzen
- **Kontrollflussorientierte Tests:**
  - Wie viele **Anweisungen** des Programms wurden abgedeckt?
  - Wie viele **Zweige** des Programms wurden abgedeckt?
  - Wie viele **Pfade** des Programms wurden abgedeckt?
  - Wie viele **Bedingungen** des Programms wurden abgedeckt?
  - ...
- **Datenflussorientierte Tests**
  - Idee: Testen verschiedener Kombinationen von Schreib- und Lesezugriffen auf Variablen

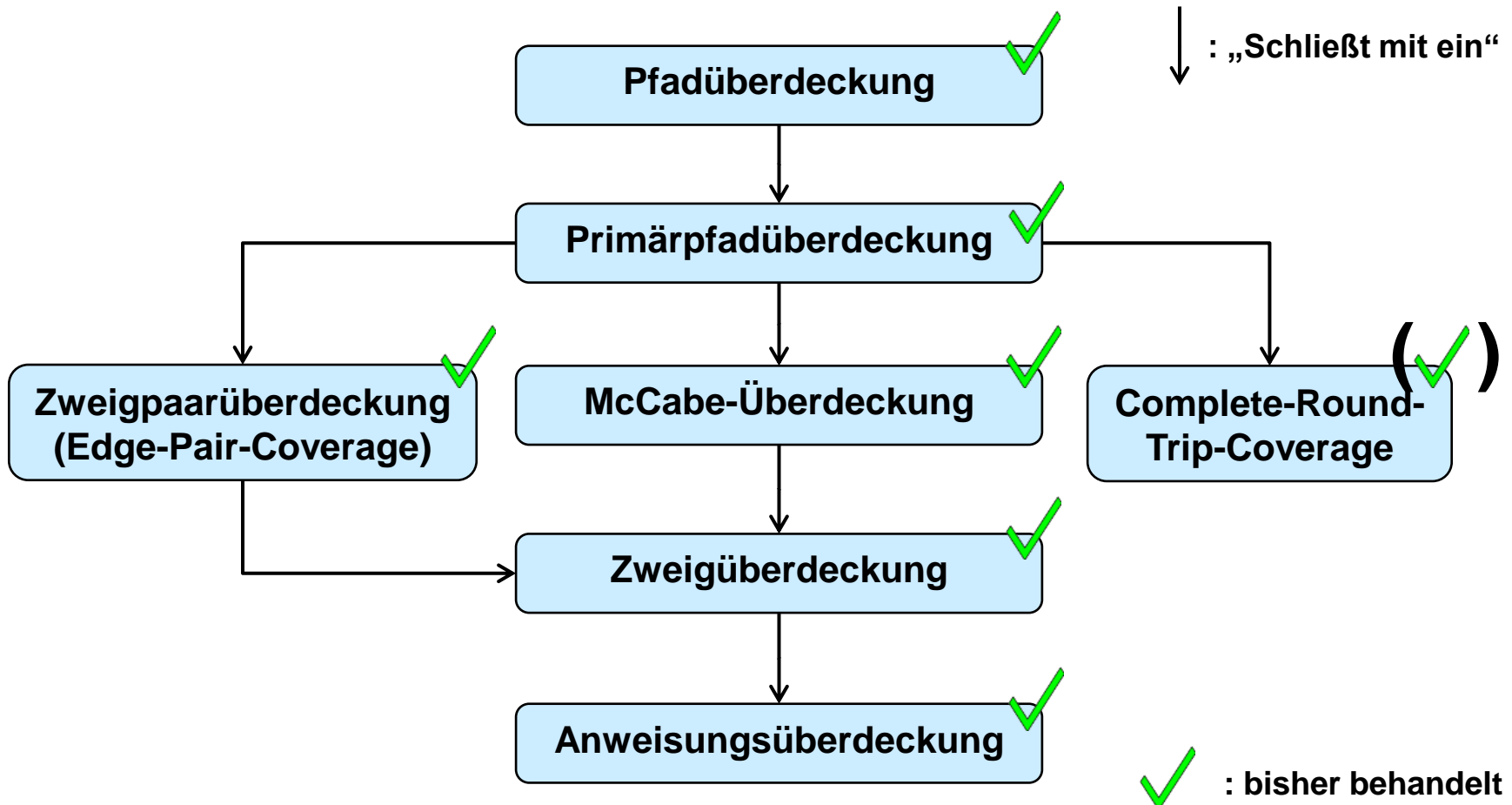
- Kontrollflussgraphen beschreiben mögliche Positionsfolgen des Programmzeigers (program counters)
- Beispiel:

```
/**  
 * calculate the Manhattan distance  
 * of two points p1 = (x1, y1), p2 = (x2, y2)  
 * by calling manhattan(x1-x2, y1-y2)  
 * @param a  
 * @param b  
 * @return |a| + |b|  
 */
```

```
public int manhattan(int a, int b){  
    if (a < 0)  
        a = -a;  
    if (b < 0)  
        b = -b;  
    return a+b;  
}
```



- Einige Überdeckungskriterien schließen andere mit ein:





# Bedingungsüberdeckung

---

- Entwickler machen oft Fehler bei komplizierten Bedingungen



# Bedingungsüberdeckung

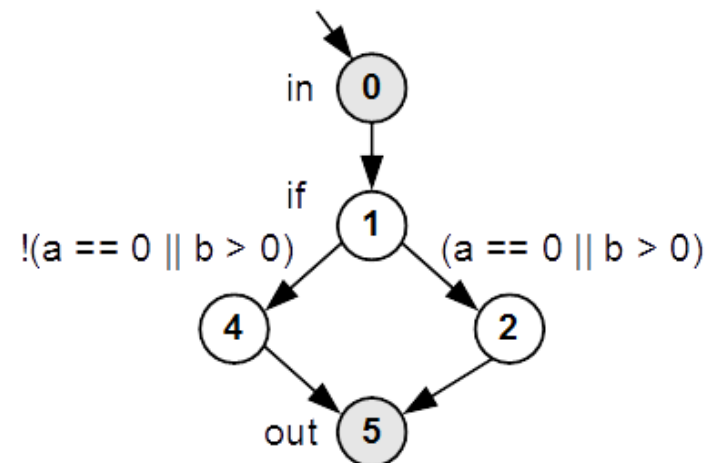
---

- Entwickler machen oft Fehler bei komplizierten Bedingungen
- Durch die kontrollflussorientierten Überdeckungskriterien wird nicht sichergestellt, dass alle diese Fehler gefunden werden



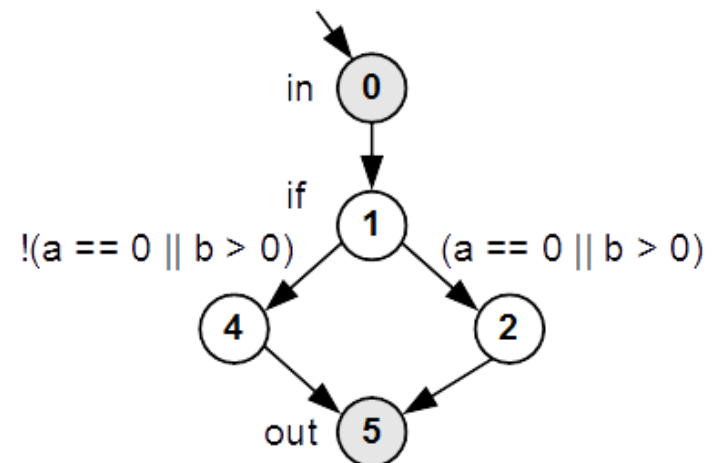
- Entwickler machen oft Fehler bei komplizierten Bedingungen
- Durch die kontrollflussorientierten Überdeckungskriterien wird nicht sichergestellt, dass alle diese Fehler gefunden werden
- **Beispiel:** Diese Tests der Methode `computeSth` führen zur **Pfadüberdeckung**: `computeSth(1,1)`, `computeSth(1,-1)`
  - Gibt es **trotzdem** einen Fehler, den wir so übersehen?

```
public int computeSth(  
    float a, float b){  
    if (a == 0 || b > 0)  
        a = b/a;  
    else  
        a = b + 2;  
    return a;  
}
```



- Entwickler machen oft Fehler bei komplizierten Bedingungen
- Durch die kontrollflussorientierten Überdeckungskriterien wird nicht sichergestellt, dass alle diese Fehler gefunden werden
- **Beispiel:** Diese Tests der Methode computeSth führen zur **Pfadüberdeckung**: computeSth(1,1), computeSth(1,-1)
  - Gibt es **trotzdem** einen Fehler, den wir so übersehen?
  - Ja, z.B. computeSth(1,0) führt zu „division by zero“!

```
public int computeSth(  
    float a, float b){  
    if (a == 0 || b > 0)  
        a = b/a;  
    else  
        a = b + 2;  
    return a;  
}
```





# Bedingungsüberdeckung

---

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen
  - **Automares Prädikat:**

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen
  - **Automares Prädikat:**
    - Einzelne Wahrheitsvariable (Boolean),

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten testen**
  - **Automares Prädikat:**
    - Einzelne Wahrheitsvariable (Boolean),
    - Vergleiche, z.B. „ $\leq$ ,  $\neq$ ,  $==$ , ...“.

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen
  - **Automares Prädikat:**
    - Einzelne Wahrheitsvariable (Boolean),
    - Vergleiche, z.B. „ $\leq$ ,  $\neq$ ,  $==$ , ...“.
    - Aufrufe Boolescher Funktionen, z.B. „.isAvailable()“



- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen
  - **Automares Prädikat:**
    - Einzelne Wahrheitsvariable (Boolean),
    - Vergleiche, z.B. „ $\leq$ ,  $\neq$ ,  $==$ , ...“.
    - Aufrufe Boolescher Funktionen, z.B. „.isAvailable()“
  - **Zusammengesetztes Prädikat:**

- **Annahme:** Entwickler machen oft Fehler bei komplizierten Bedingungen
- **Idee:** Nicht nur beide Zweige von Bedingungen (z.B. Fallunterscheidungs- oder Schleifenbedingungen) testen, **sondern verschiedene Auswertungen von atomaren Prädikaten und zusammengesetzten Prädikaten** testen
  - **Atomares Prädikat:**
    - Einzelne Wahrheitsvariable (Boolean),
    - Vergleiche, z.B. „ $\leq$ ,  $\neq$ ,  $==$ , ...“.
    - Aufrufe Boolescher Funktionen, z.B. „`isAvailable()`“
  - **Zusammengesetztes Prädikat:**
    - Durch logische Junktoren zusammengesetzt, z.B. „ $(a \ \&\& \ b)$ “, „ $!a$ “

- **Einfache Bedingungsüberdeckung:**  
Alle atomaren Prädikate nehmen  
mindestens einmal beide  
Wahrheitswerte an



# Bedingungsüberdeckung

---

**if** ((a && !b) || (!a && b)) ...

- **Einfache Bedingungsüberdeckung:**  
Alle atomaren Prädikate nehmen  
mindestens einmal beide  
Wahrheitswerte an



# Bedingungsüberdeckung

---

**if** ((a && !b) || (!a && b)) ...

- **Einfache Bedingungsüberdeckung:**

Alle atomaren Prädikate nehmen  
mindestens einmal beide  
Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

```
if ((a && !b) || (!a && b)) ...
```

- **Einfache Bedingungsüberdeckung:**

Alle atomaren Prädikate nehmen  
mindestens einmal beide  
Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

- **Minimale Bedingungsüberdeckung:**

Alle atomaren und zusammengesetzten  
Prädikate nehmen mindestens einmal  
beide Wahrheitswerte an

**if** ((a && !b) || (!a && b)) ...

- **Einfache Bedingungsüberdeckung:**

Alle atomaren Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

- **Minimale Bedingungsüberdeckung:**

Alle atomaren und zusammengesetzten Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

T3: a = false, b = false

(nur T3 führt dazu, dass die Disjunktion insgesamt false ergibt)

**if** ((a && !b) || (!a && b)) ...

- **Einfache Bedingungsüberdeckung:**

Alle atomaren Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

- **Minimale Bedingungsüberdeckung:**

Alle atomaren und zusammengesetzten Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

T3: a = false, b = false

(nur T3 führt dazu, dass die Disjunktion insgesamt false ergibt)

- **Mehrfache Bedingungsüberdeckung:**

Alle Kombinationen von Auswertungen der atomaren Prädikate getestet



**if** ((a && !b) || (!a && b)) ...

- **Einfache Bedingungsüberdeckung:**

Alle atomaren Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

- **Minimale Bedingungsüberdeckung:**

Alle atomaren und zusammengesetzten Prädikate nehmen mindestens einmal beide Wahrheitswerte an

T1: a = false, b = true

T2: a = true, b = false

T3: a = false, b = false

(nur T3 führt dazu, dass die Disjunktion insgesamt false ergibt)

- **Mehrfache Bedingungsüberdeckung:**

Alle Kombinationen von Auswertungen der atomaren Prädikate getestet

T1: a = false, b = true

T2: a = true, b = false

T3: a = false, b = false

T4: a = true, b = true

# Datenflussorientierte Überdeckungskriterien – Defs-Uses-Überdeckungskriterien

---

- Bisherige Überdeckungskriterien basierten auf dem **Kontrollfluss bzw. den Bedingungen**

# Datenflussorientierte Überdeckungskriterien – Defs-Uses-Überdeckungskriterien

---

- Bisherige Überdeckungskriterien basierten auf dem **Kontrollfluss bzw. den Bedingungen**
- Im Folgenden betrachten wir eine Reihe von Überdeckungskriterien, die sich auf den **Datenfluss** beziehen

# Datenflussorientierte Überdeckungskriterien – Defs-Uses-Überdeckungskriterien

---

- Bisherige Überdeckungskriterien basierten auf dem **Kontrollfluss bzw. den Bedingungen**
- Im Folgenden betrachten wir eine Reihe von Überdeckungskriterien, die sich auf den **Datenfluss** beziehen
- **Idee:** Bei der Ausführung eines Programms werden Variablen geschrieben und gelesen...



# Datenflussorientierte Überdeckungskriterien

## – Defs-Uses-Überdeckungskriterien

---

- Bisherige Überdeckungskriterien basierten auf dem **Kontrollfluss bzw. den Bedingungen**
- Im Folgenden betrachten wir eine Reihe von Überdeckungskriterien, die sich auf den **Datenfluss** beziehen
- **Idee:** Bei der Ausführung eines Programms werden Variablen geschrieben und gelesen...
  - bei verschiedenen Folgen von Schreib- und Lesezugriffen verhält sich das Programm andersartig



# Datenflussorientierte Überdeckungskriterien

## – Defs-Uses-Überdeckungskriterien

---

- Bisherige Überdeckungskriterien basierten auf dem **Kontrollfluss bzw. den Bedingungen**
- Im Folgenden betrachten wir eine Reihe von Überdeckungskriterien, die sich auf den **Datenfluss** beziehen
- **Idee:** Bei der Ausführung eines Programms werden Variablen werden geschrieben und gelesen...
  - bei verschiedenen Folgen von Schreib- und Lesezugriffen verhält sich das Programm andersartig
  - **Hinter verschiedenen Folgen von Schreib- und Lesezugriffen können verschiedene Fehler verborgen sein**



# Defs-Uses-Überdeckungskriterien

---



# Defs-Uses-Überdeckungskriterien

---

- **Definition** einer Variable  
(def)
  - Variable wird Wert zugewiesen





# Defs-Uses-Überdeckungskriterien

---

- **Definition** einer Variable (def)
  - Variable wird Wert zugewiesen
- **Lesender Zugriff** auf eine Variable (**use**)



# Defs-Uses-Überdeckungskriterien

---

- **Definition** einer Variable (**def**)
  - Variable wird Wert zugewiesen
- **Lesender Zugriff** auf eine Variable (**use**)
  - **Berechnende** Nutzung (**c-use**)
    - Variable wird in Berechnung oder Zuweisung gelesen

- **Definition** einer Variable (def)
  - Variable wird Wert zugewiesen
- **Lesender Zugriff** auf eine Variable (**use**)
  - **Berechnende** Nutzung (c-use)
    - Variable wird in Berechnung oder Zuweisung gelesen
  - **Prädikative** Nutzung (p-use)
    - Variable wird in Bedingung gelesen

- **Definition** einer Variable (**def**)

- Variable wird Wert zugewiesen

- **Lesender Zugriff** auf eine Variable (**use**)

- **Berechnende Nutzung** (**c-use**)

- Variable wird in Berechnung oder Zuweisung gelesen

- **Prädikative Nutzung** (**p-use**)

- Variable wird in Bedingung gelesen

```
public static float pow(float x, int n){  
    float result;  
    int i;  
  
    if(n < 0)  
        i = -n;  
    else  
        i = n;  
  
    result = 1;  
    while(i > 0){  
        result *= x;  
        i--;  
    }  
  
    if (n < 0)  
        result = 1 / result;  
  
    return result;  
}
```



# Defs-Uses-Überdeckungskriterien

- **Definition** einer Variable (**def**)

- Variable wird Wert zugewiesen

- **Lesender Zugriff** auf eine Variable (**use**)

- **Berechnende Nutzung** (**c-use**)

- Variable wird in Berechnung oder Zuweisung gelesen

- **Prädikative Nutzung** (**p-use**)

- Variable wird in Bedingung gelesen

```
public static float pow(float x, int n){  
    float result;  
    int i;  
  
    if(n < 0)  
        i = -n;  
    else  
        i = n;  
  
    result = 1;  
    while(i > 0){  
        result *= x;  
        i--;  
    }  
  
    if (n < 0)  
        result = 1 / result;  
  
    return result;  
}
```

**Frage:** wie viele defs, c-uses und p-uses?



# Defs-Uses-Überdeckungskriterien

```
public static float pow(float x, int n){  
    float result;  
    int i;  
  
    if(n < 0)  
        i = -n;  
    else  
        i = n;  
  
    result = 1;  
    while(i > 0){  
        result *= x;  
        i--;  
    }  
  
    if (n < 0)  
        result = 1 / result;  
  
    return result;  
}
```

```
public static float pow(float x, int n){  
    float result;  
    int i;  
  
    if(n < 0)  
        i = -n;  
    else  
        i = n;  
  
    result = 1;  
    while(i > 0){  
        result *= x;  
        i--;  
    }  
  
    if (n < 0)  
        result = 1 / result;  
  
    return result;  
}
```



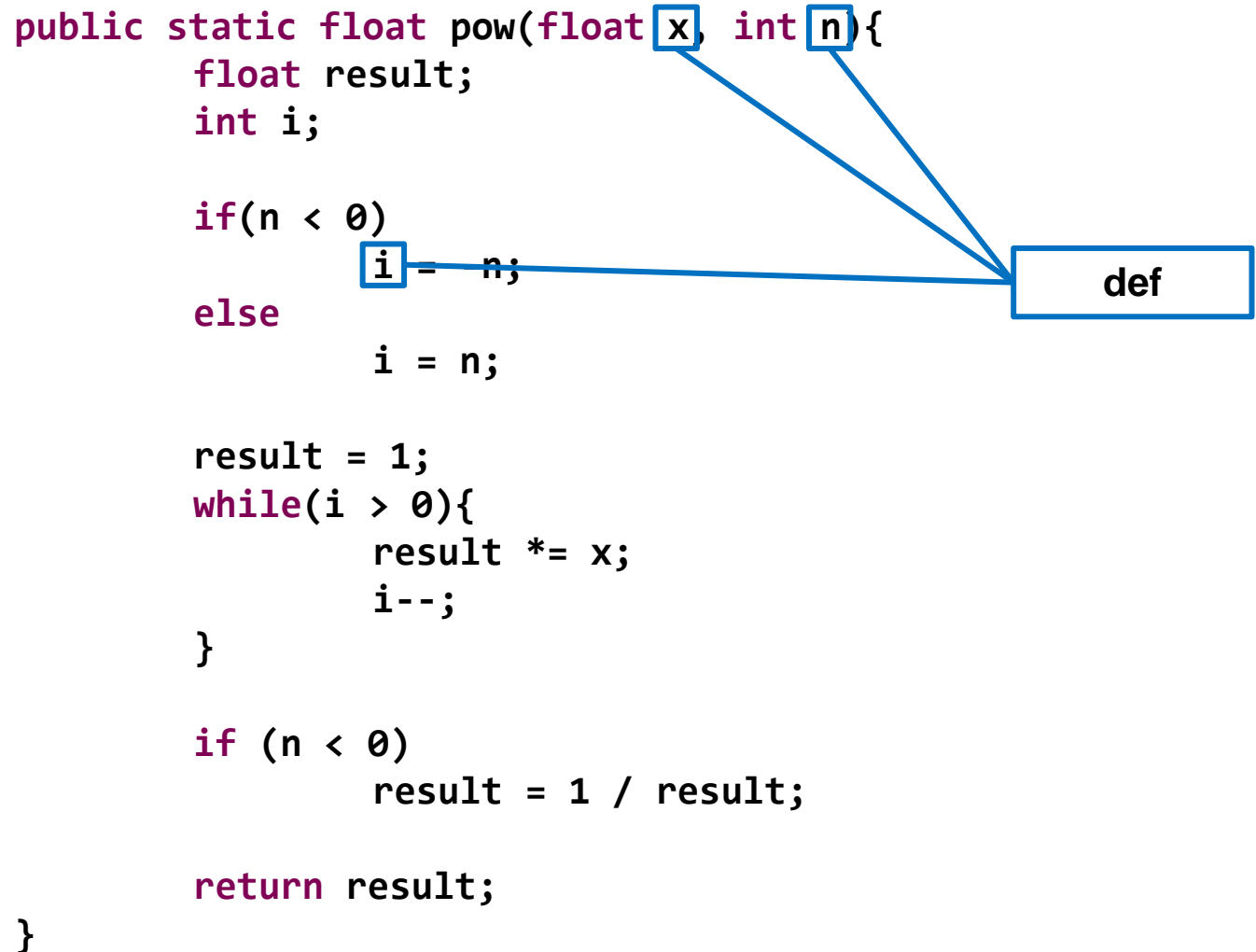
def

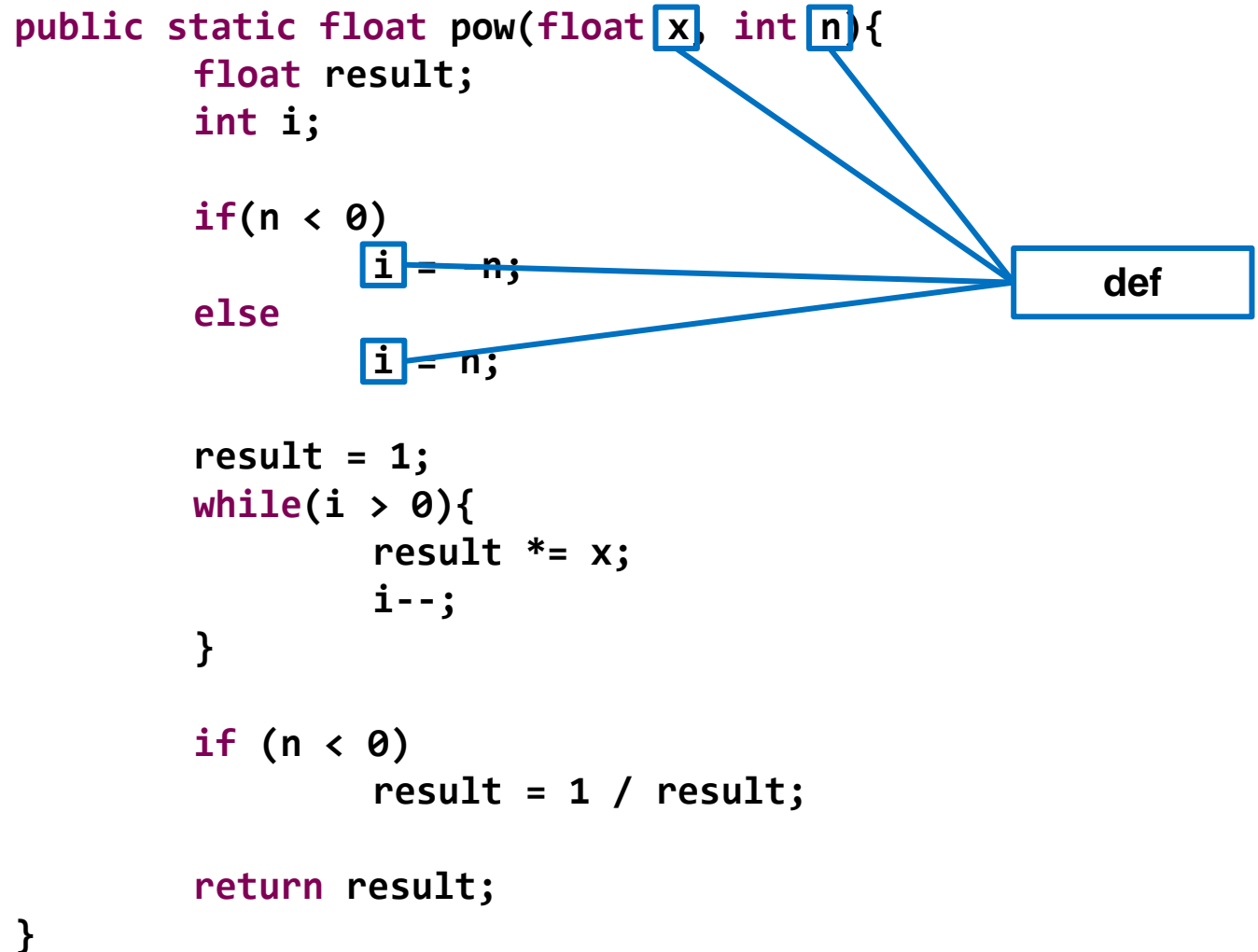
```
public static float pow(float x, int n){  
    float result;  
    int i;  
  
    if(n < 0)  
        i = -n;  
    else  
        i = n;  
  
    result = 1;  
    while(i > 0){  
        result *= x;  
        i--;  
    }  
  
    if (n < 0)  
        result = 1 / result;  
  
    return result;  
}
```

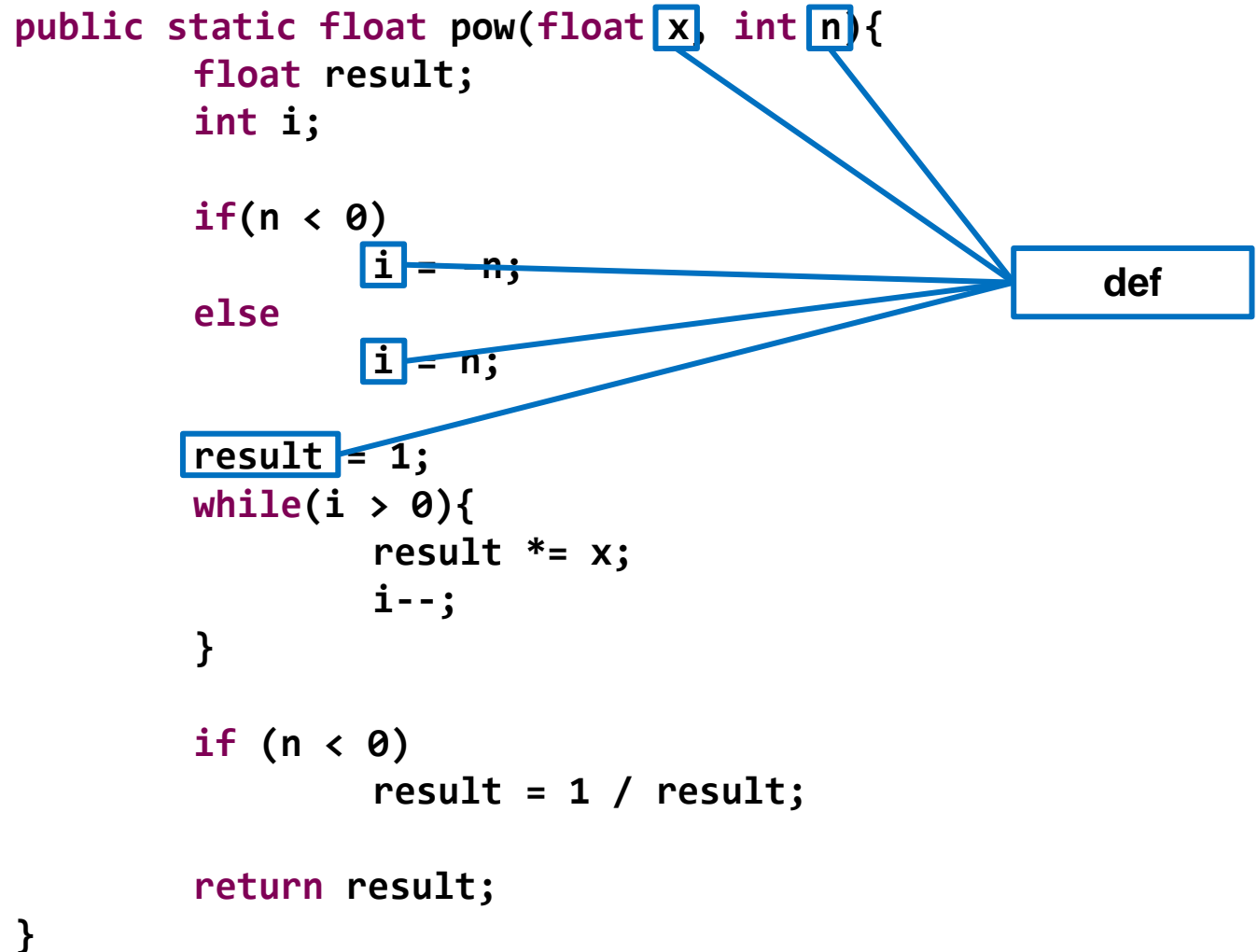


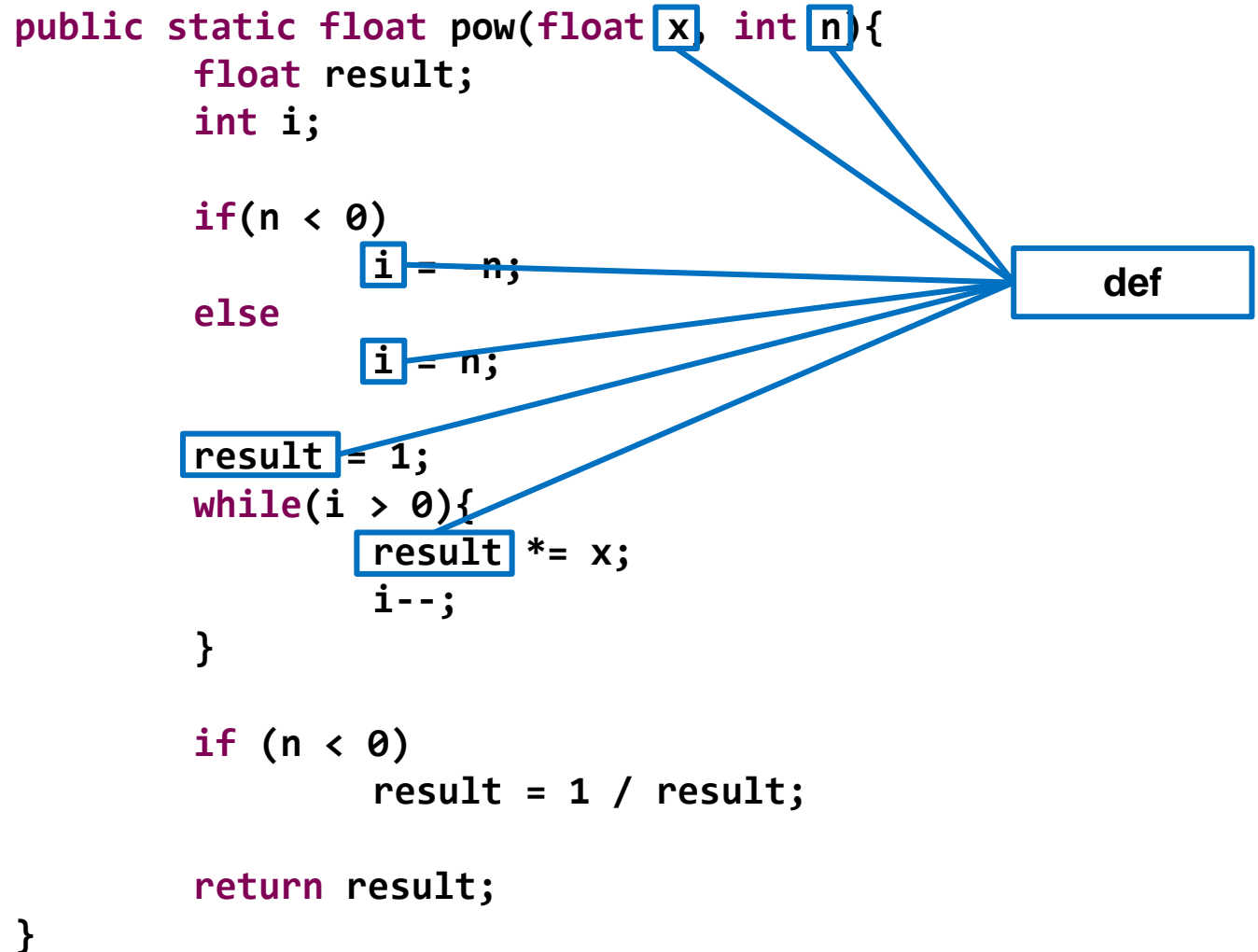
def

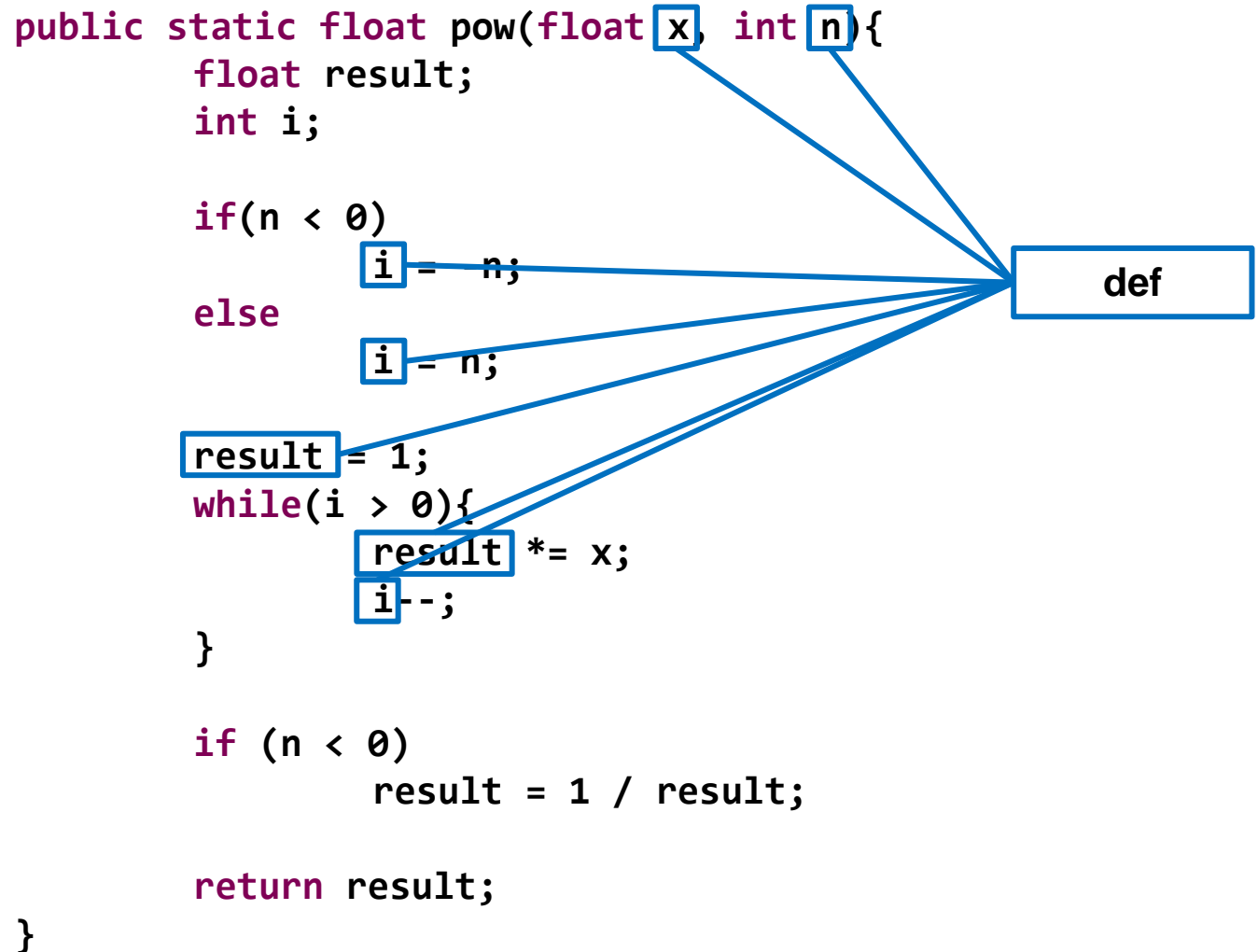


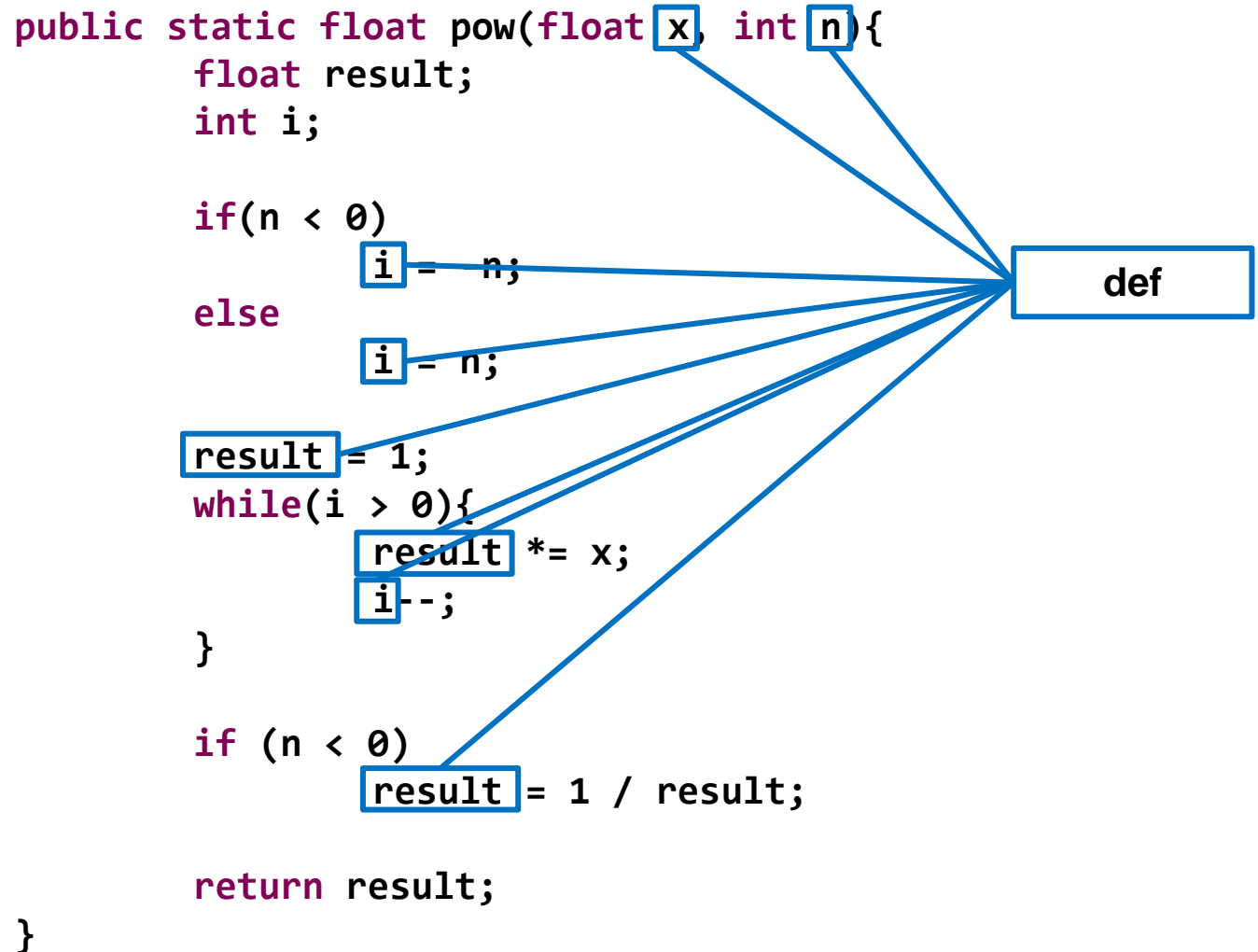


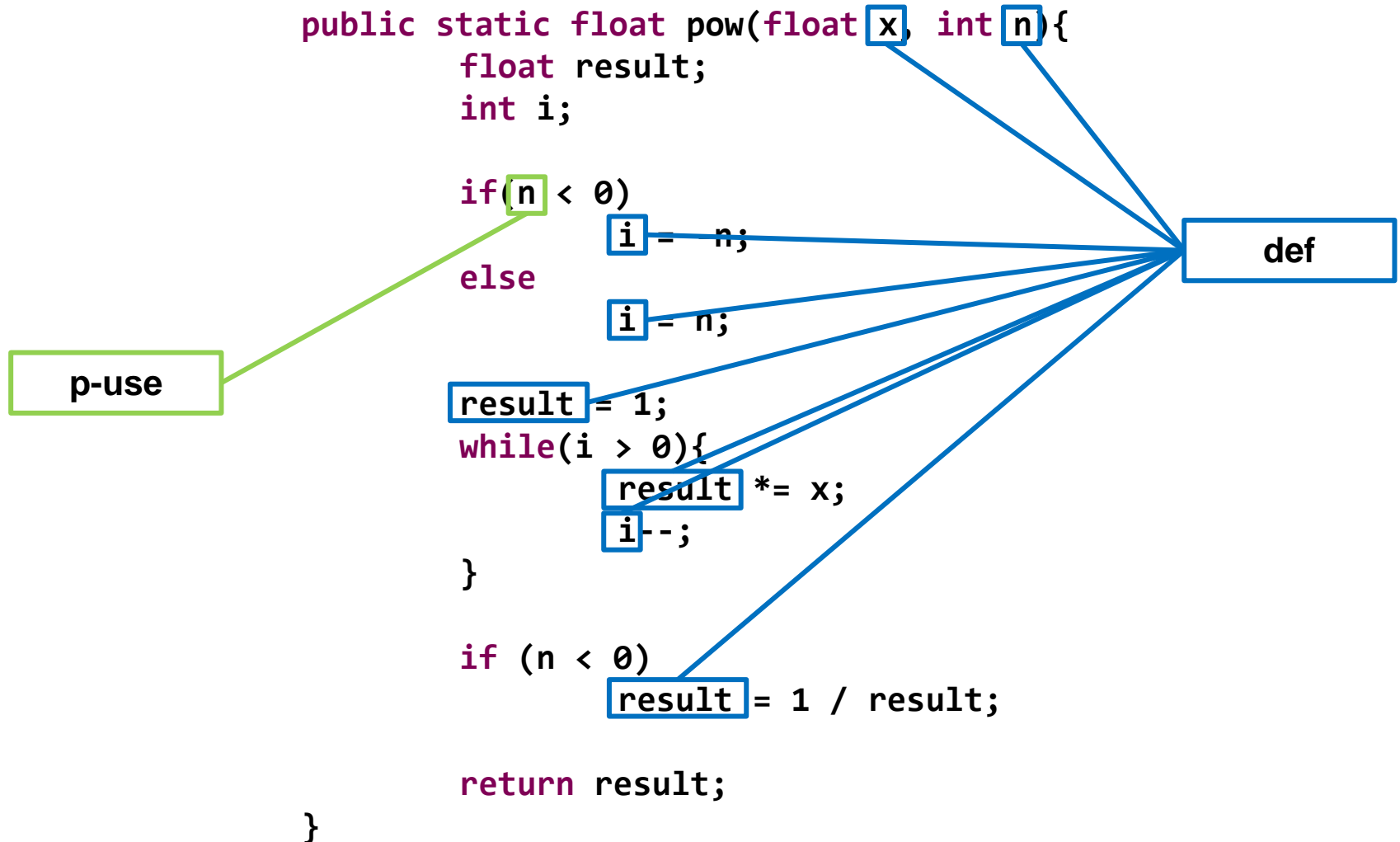


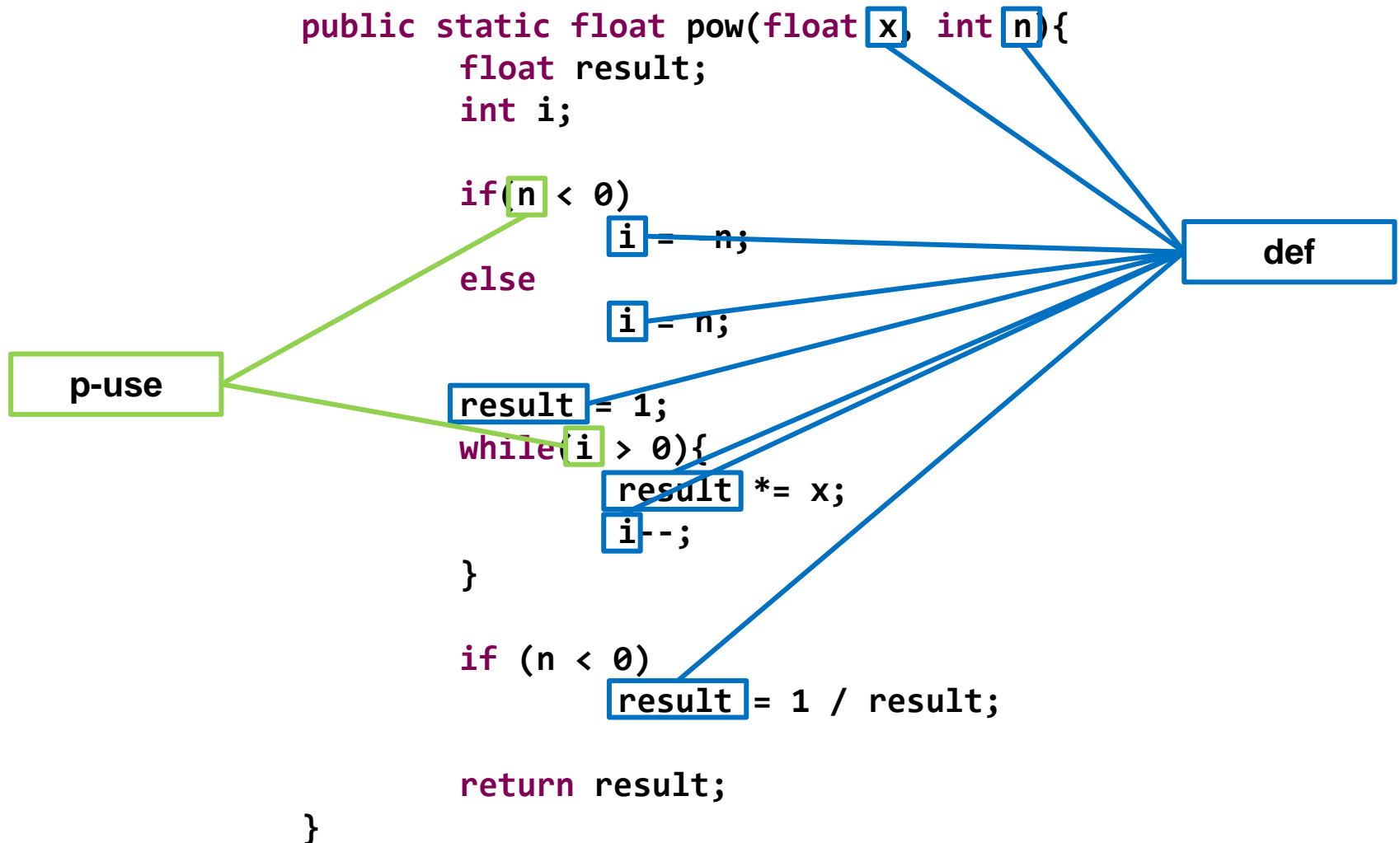




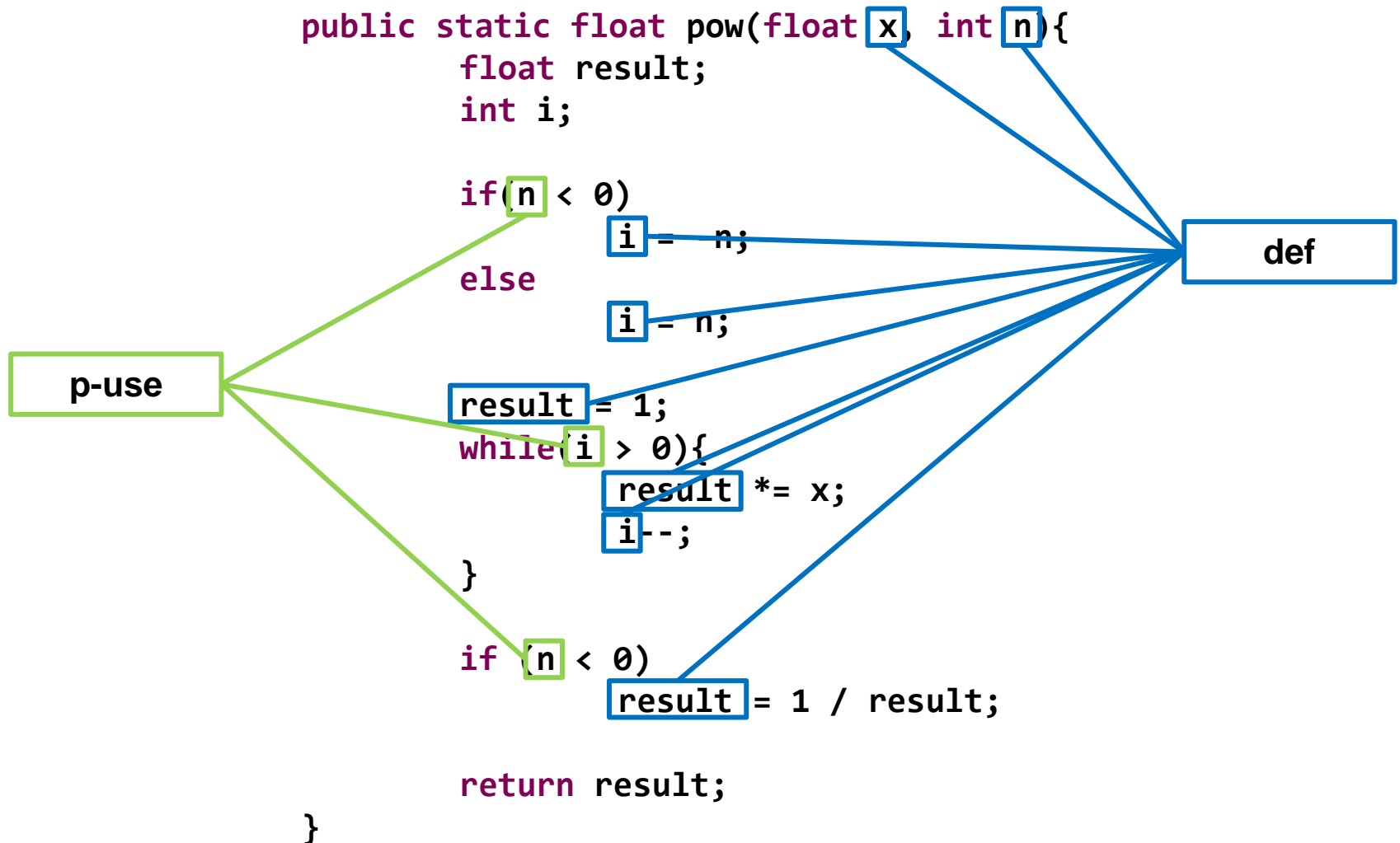


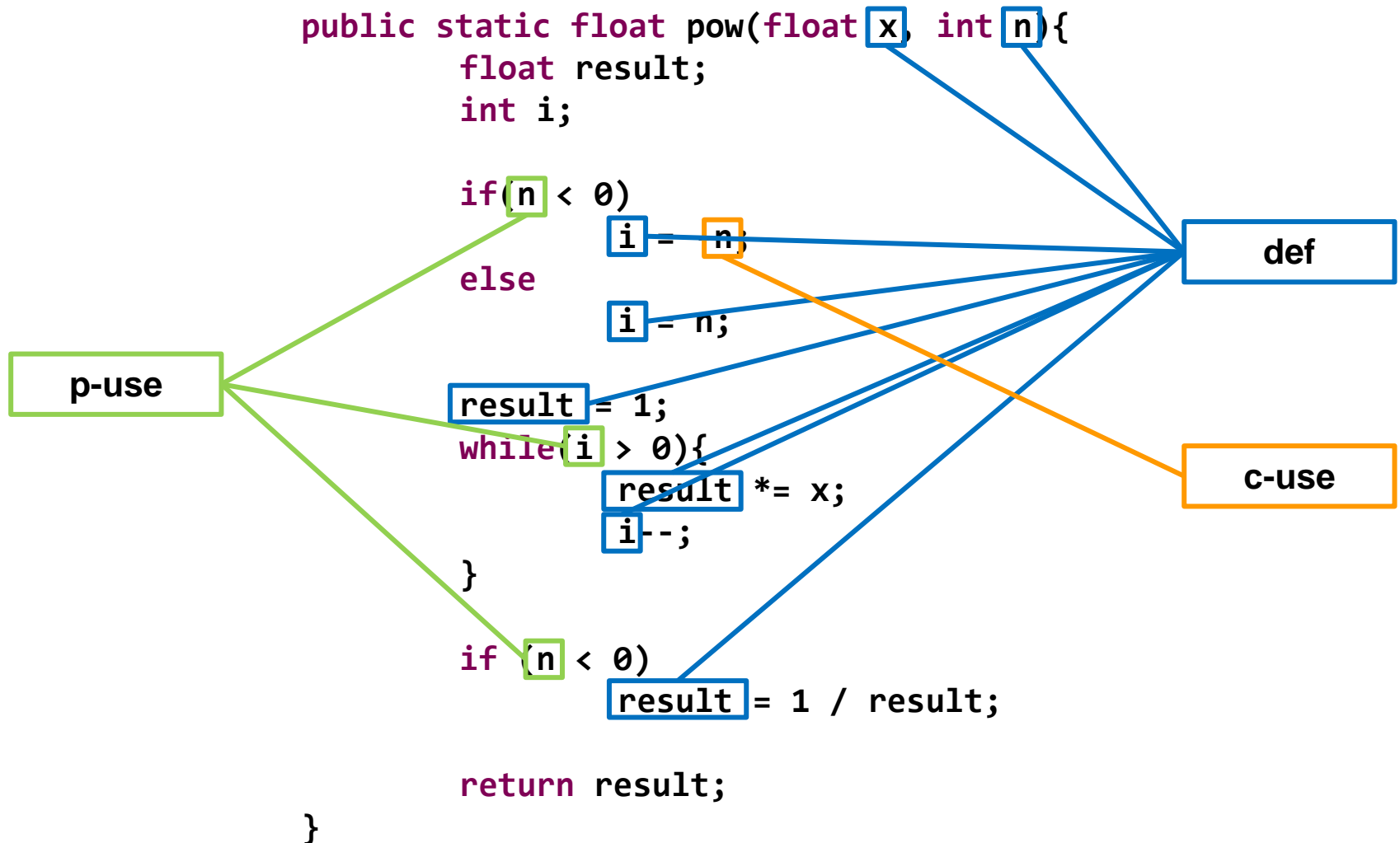




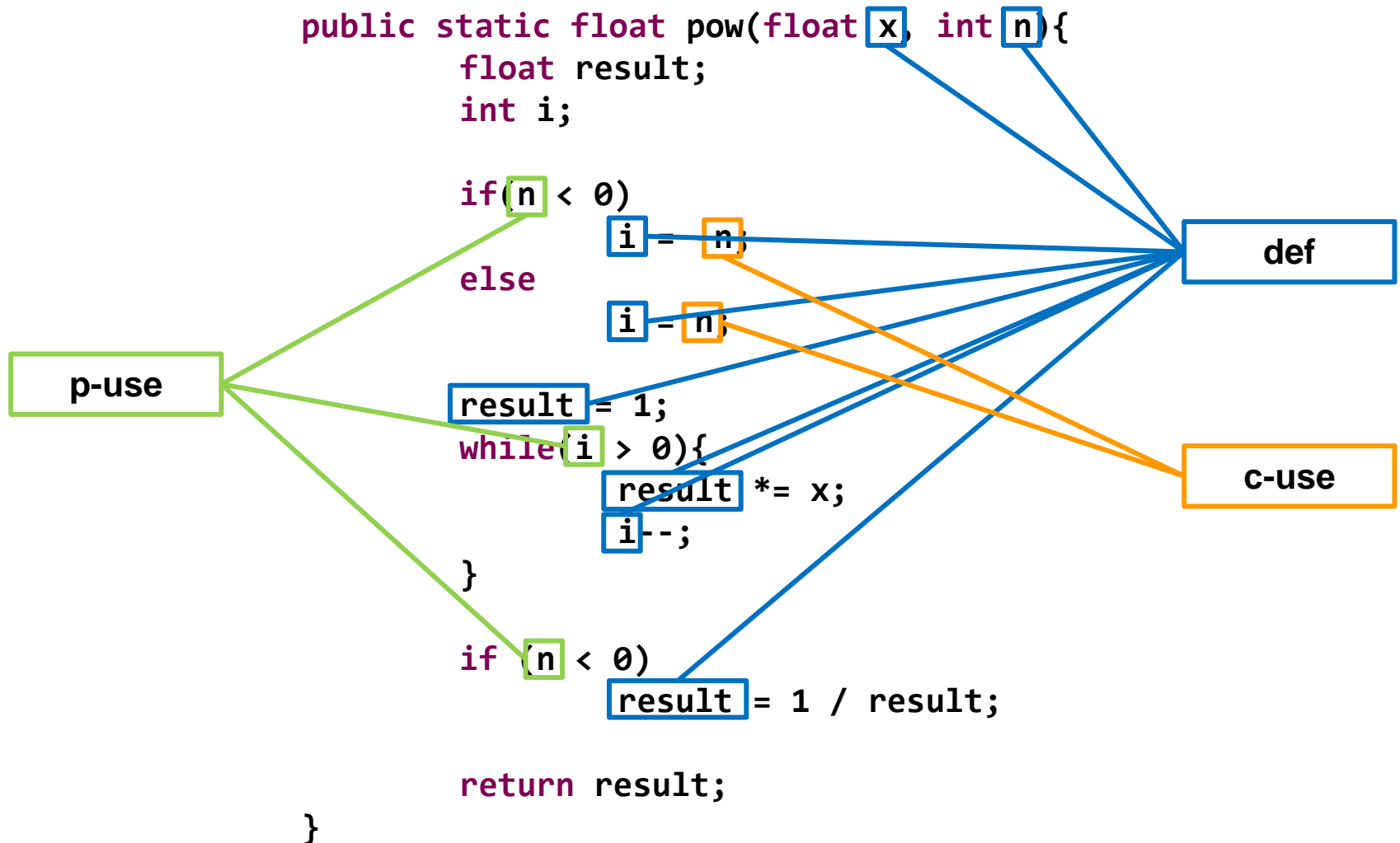


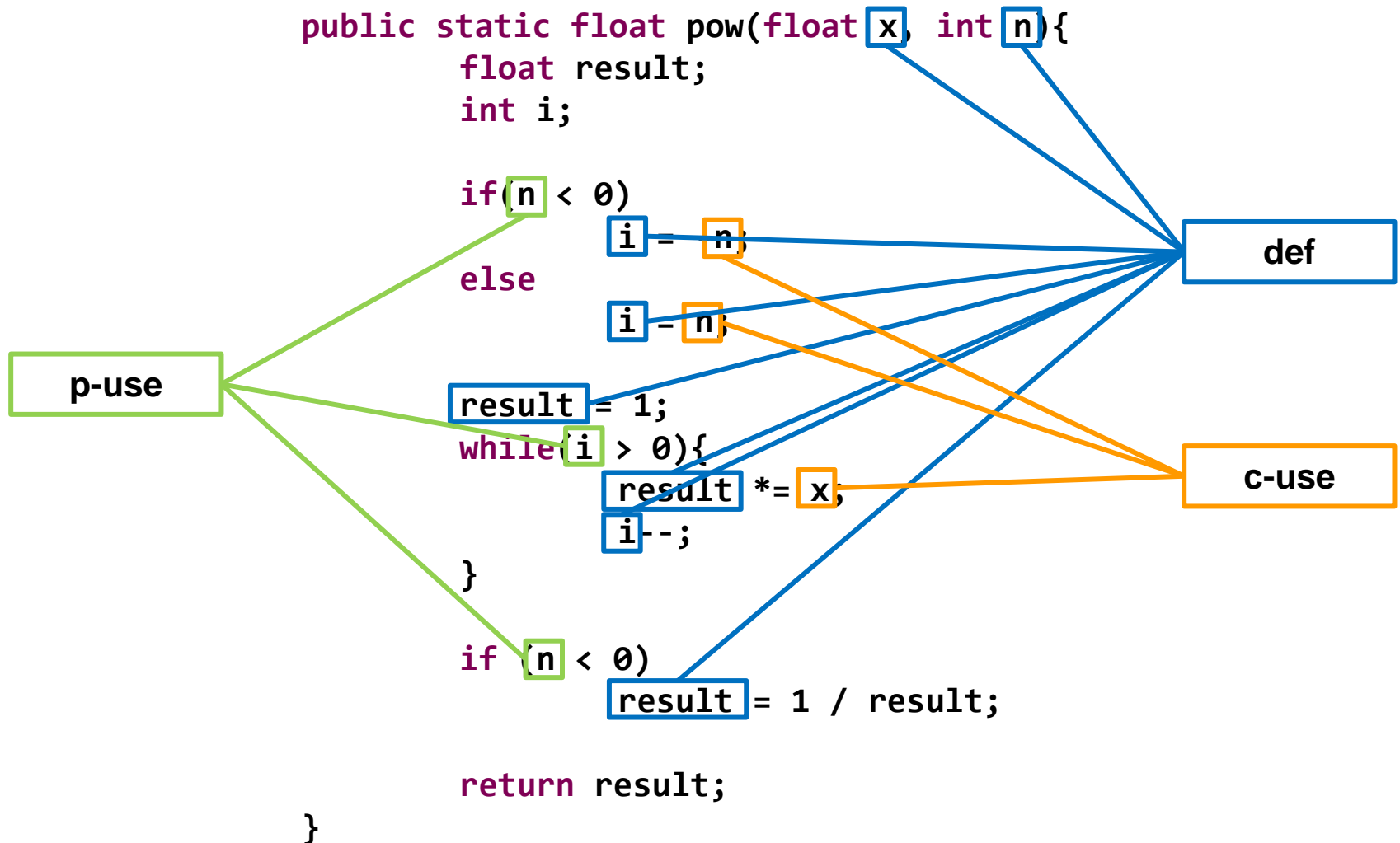


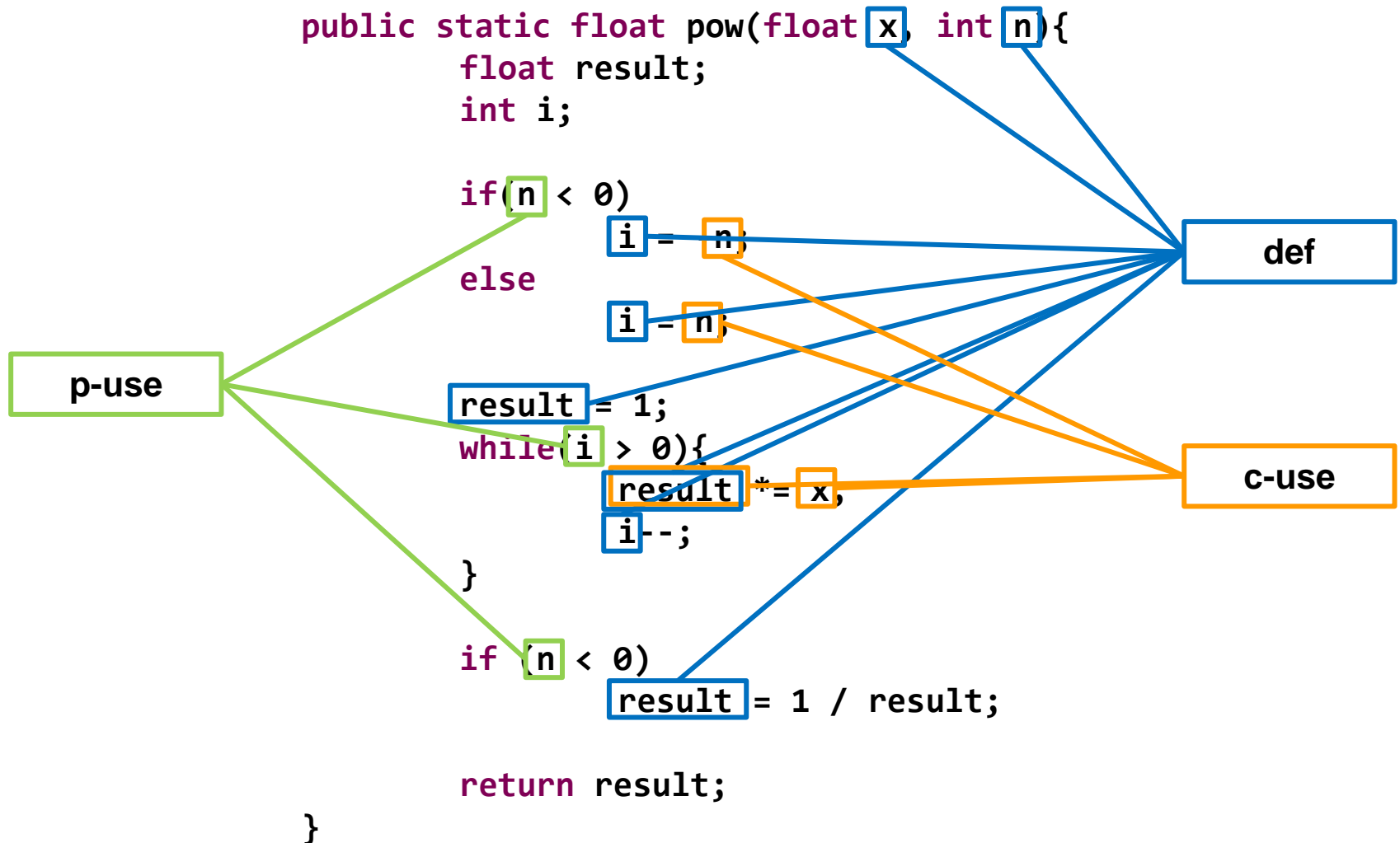


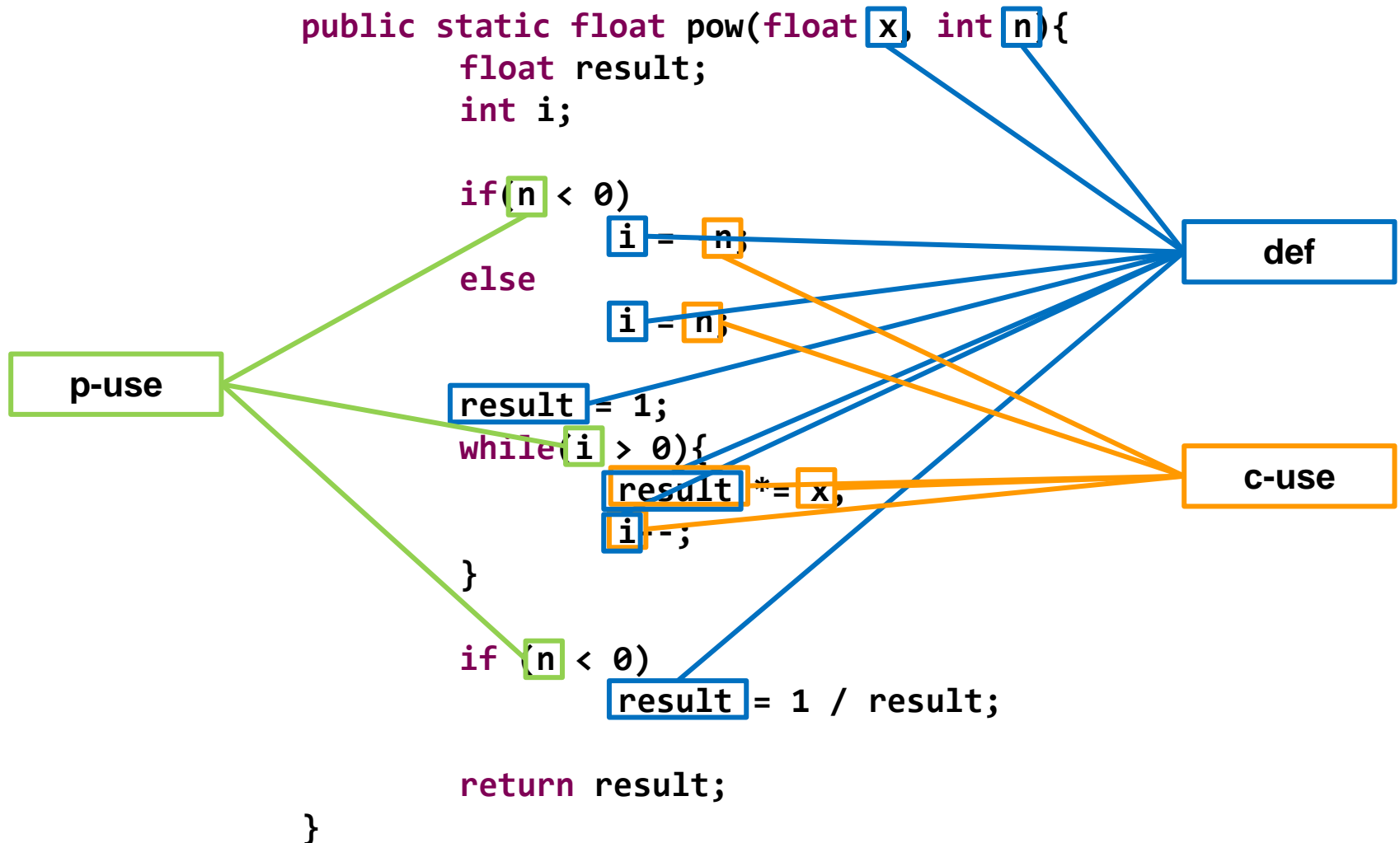


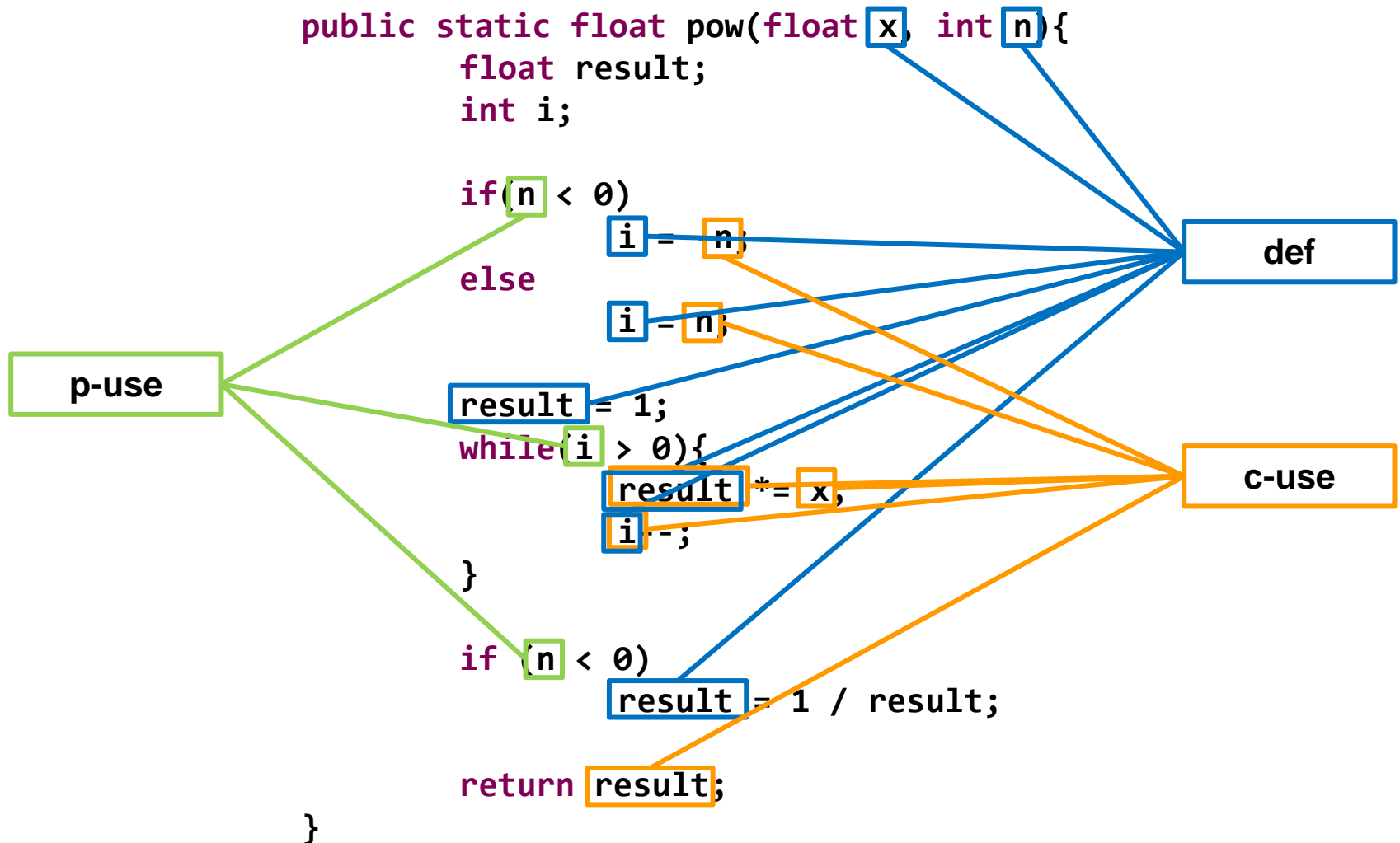
# Defs-Uses-Überdeckungskriterien

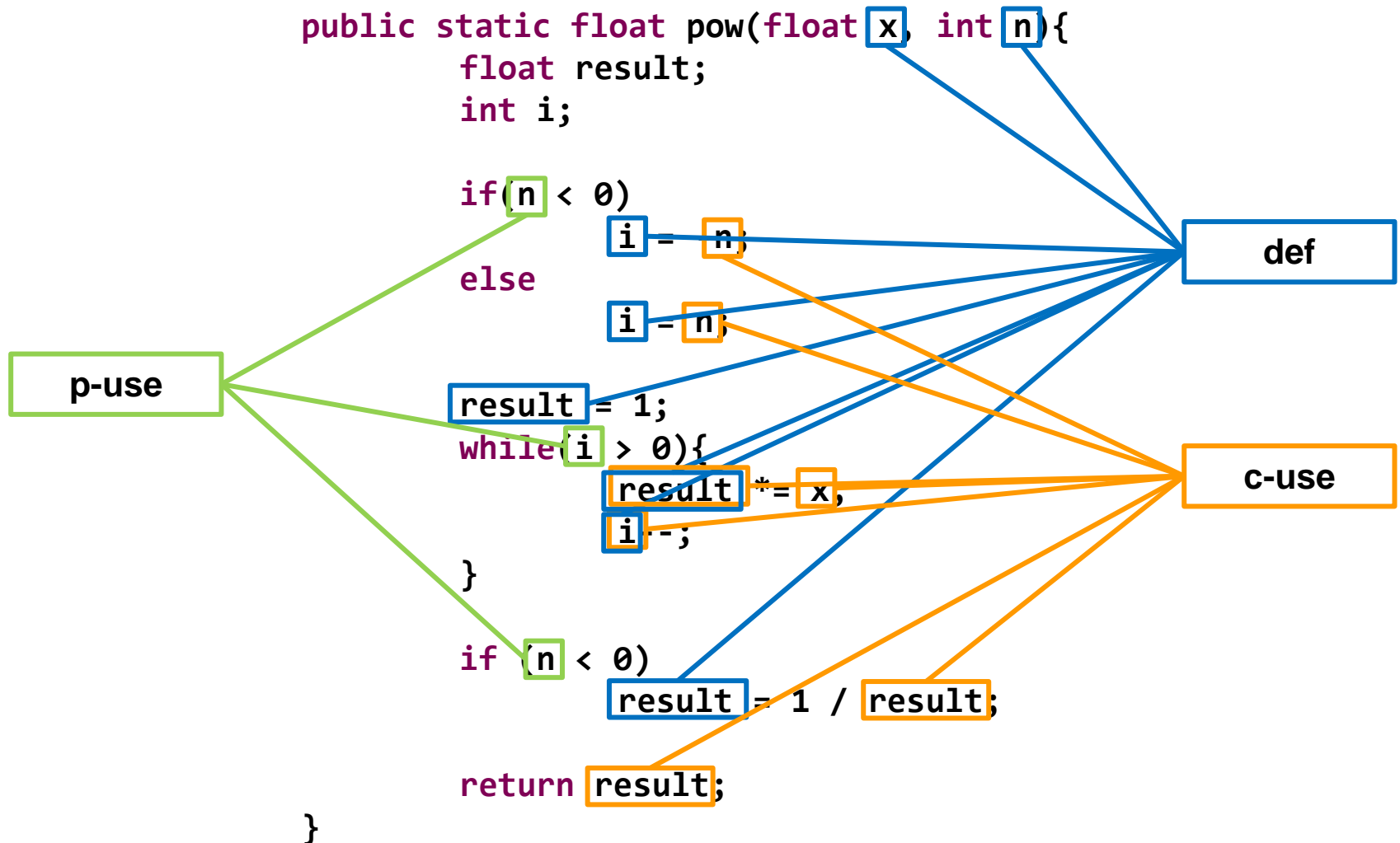




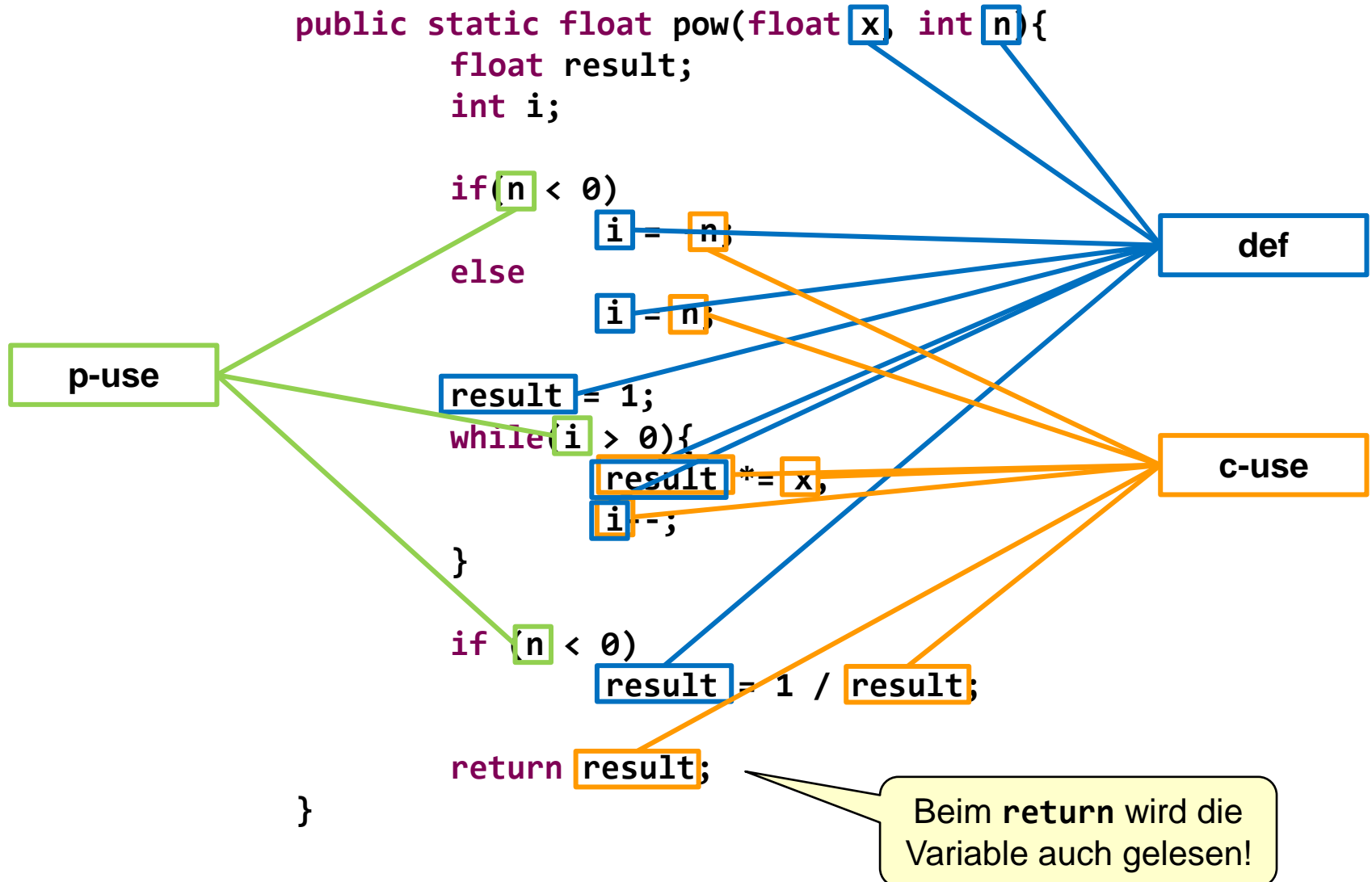




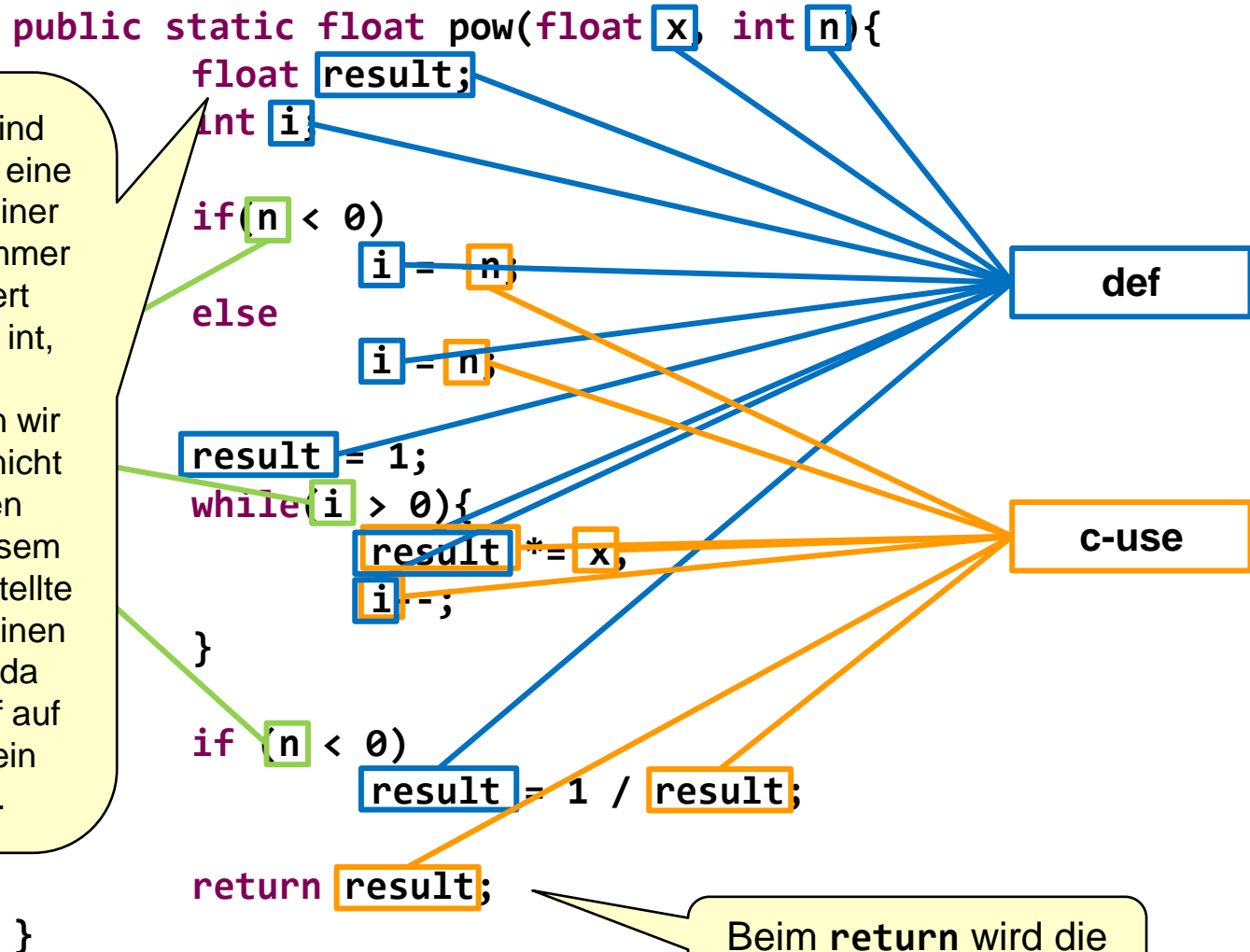




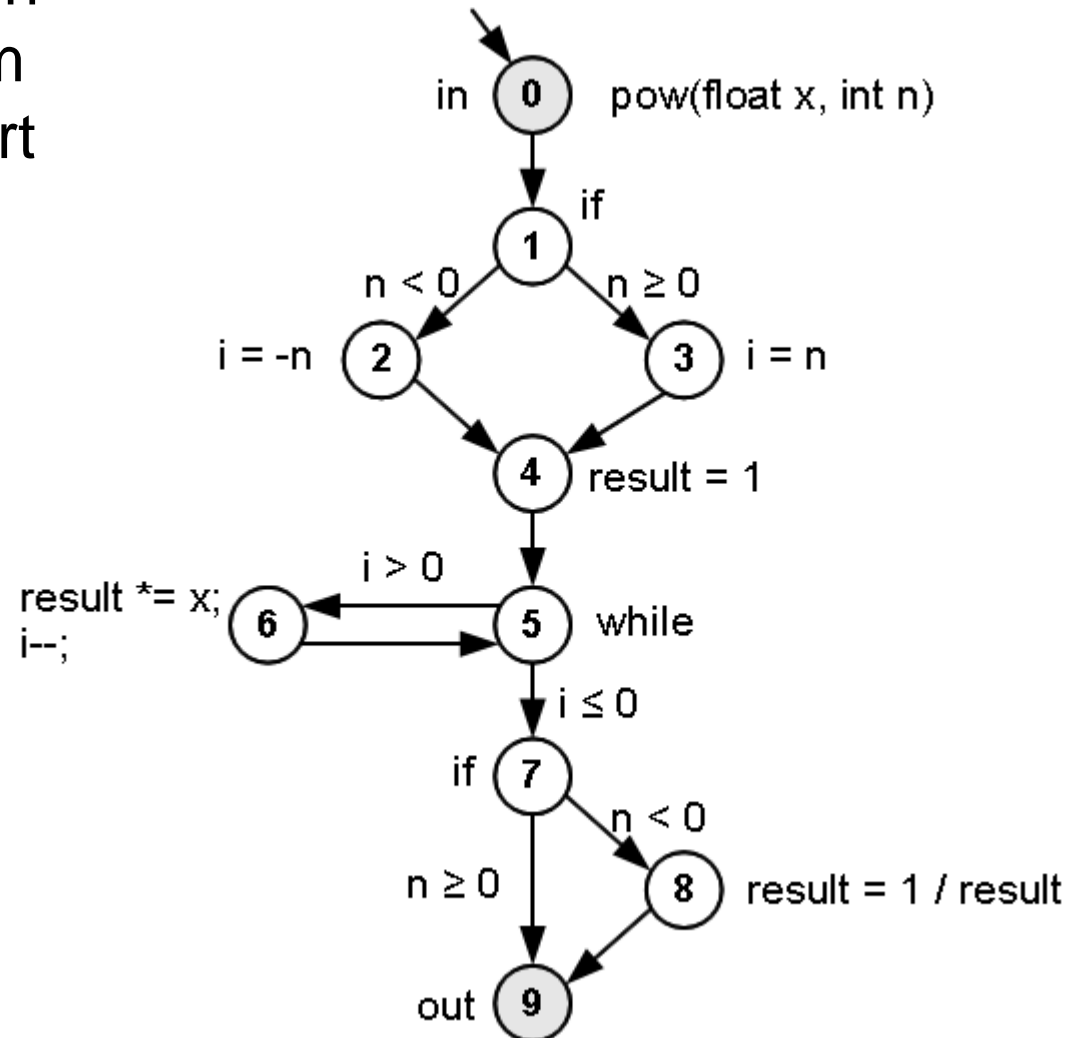




Streng genommen sind dies auch defs, da die eine Variable in Java bei einer solchen Deklaration immer mit dem Default-Wert initialisiert wird (0 bei int, 0.0 bei float). Im Folgenden betrachten wir diese Stellen jedoch nicht als defs. Wir werden sehen, dass es in diesem Beispiel für die vorgestellte Testmethodik auch keinen Unterschied macht, da keine lesender Zugriff auf die Variablen ohne ein weiteres def. folgt.

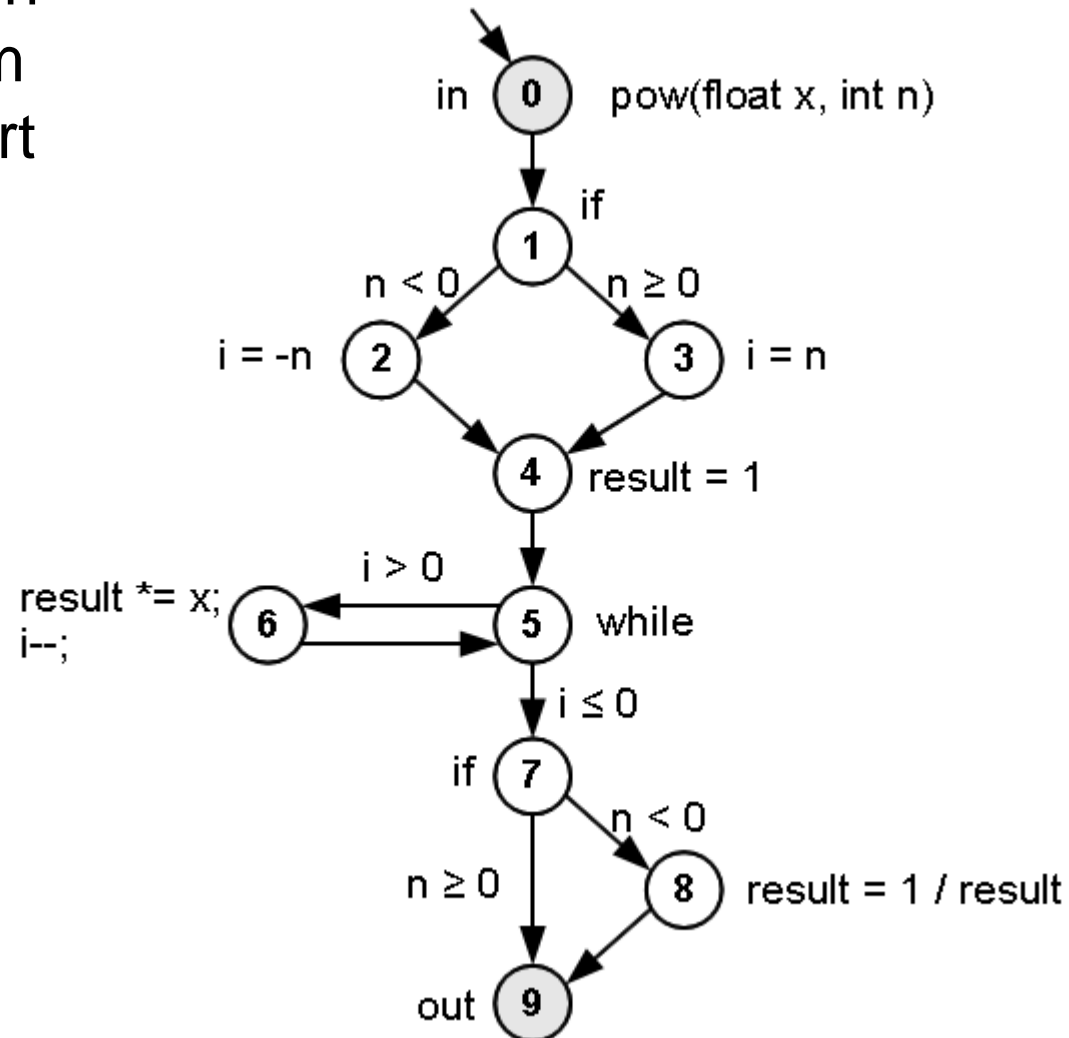


- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:



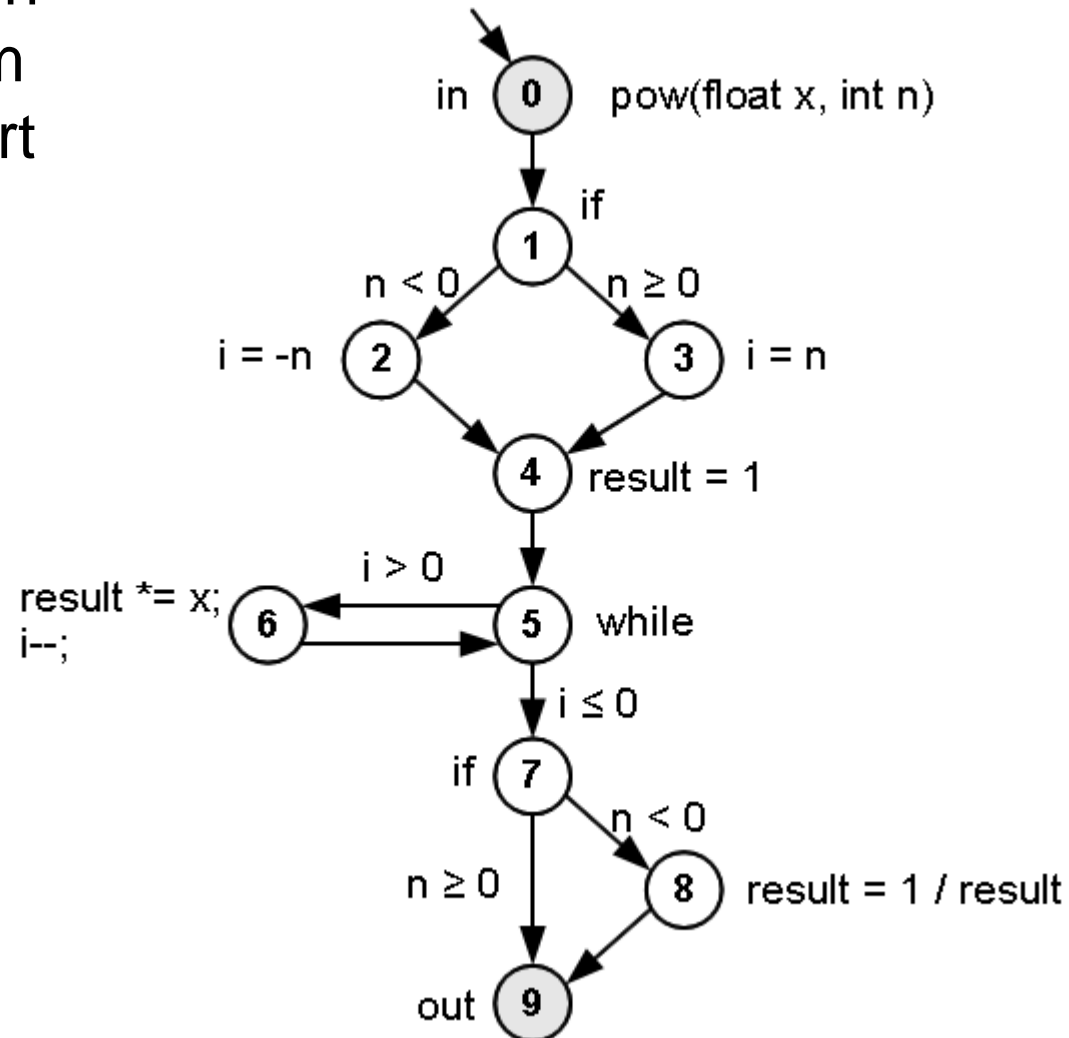
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten  $n$  definierten Variablen



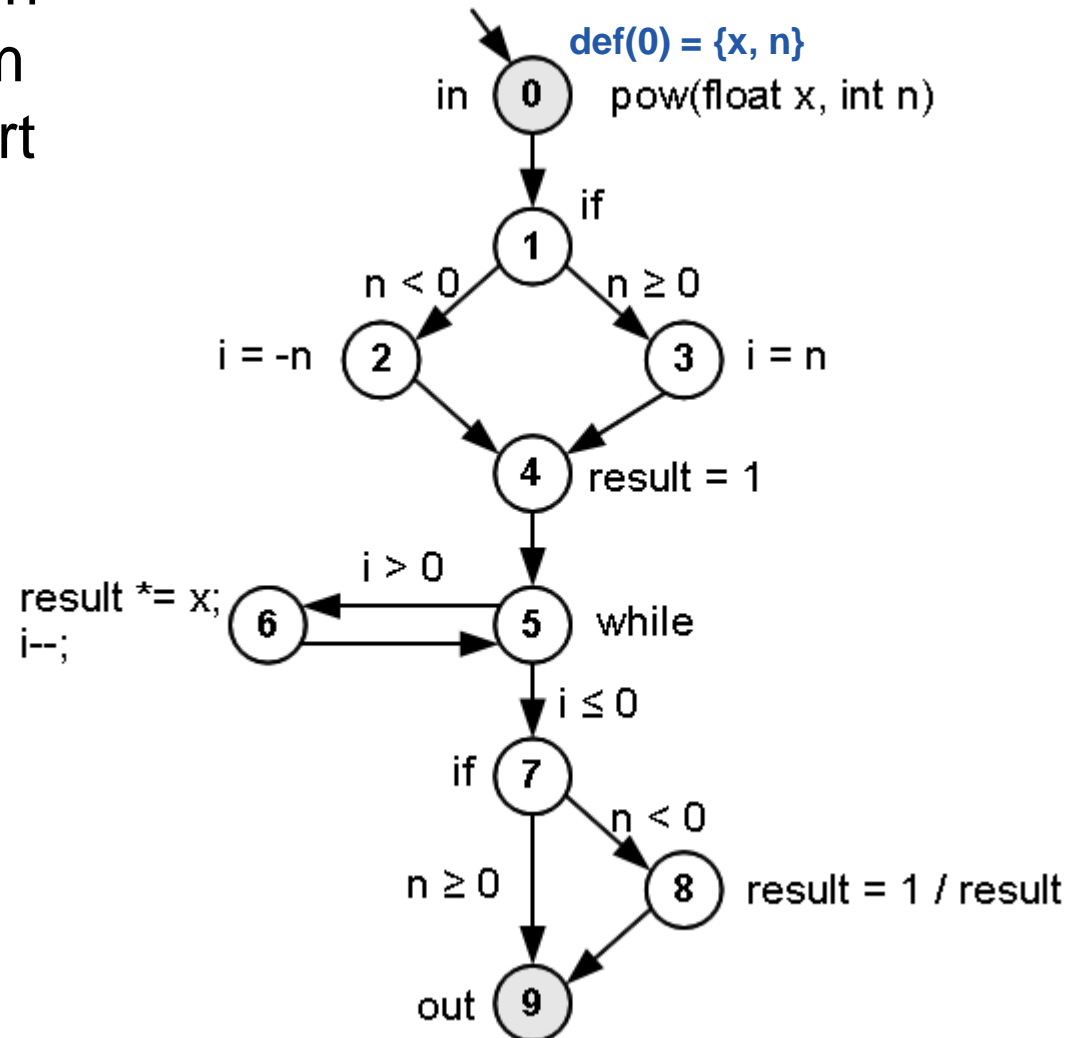
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten  $n$  definierten Variablen
- **use(n)**: die in Knoten  $n$  gelesenen Variablen



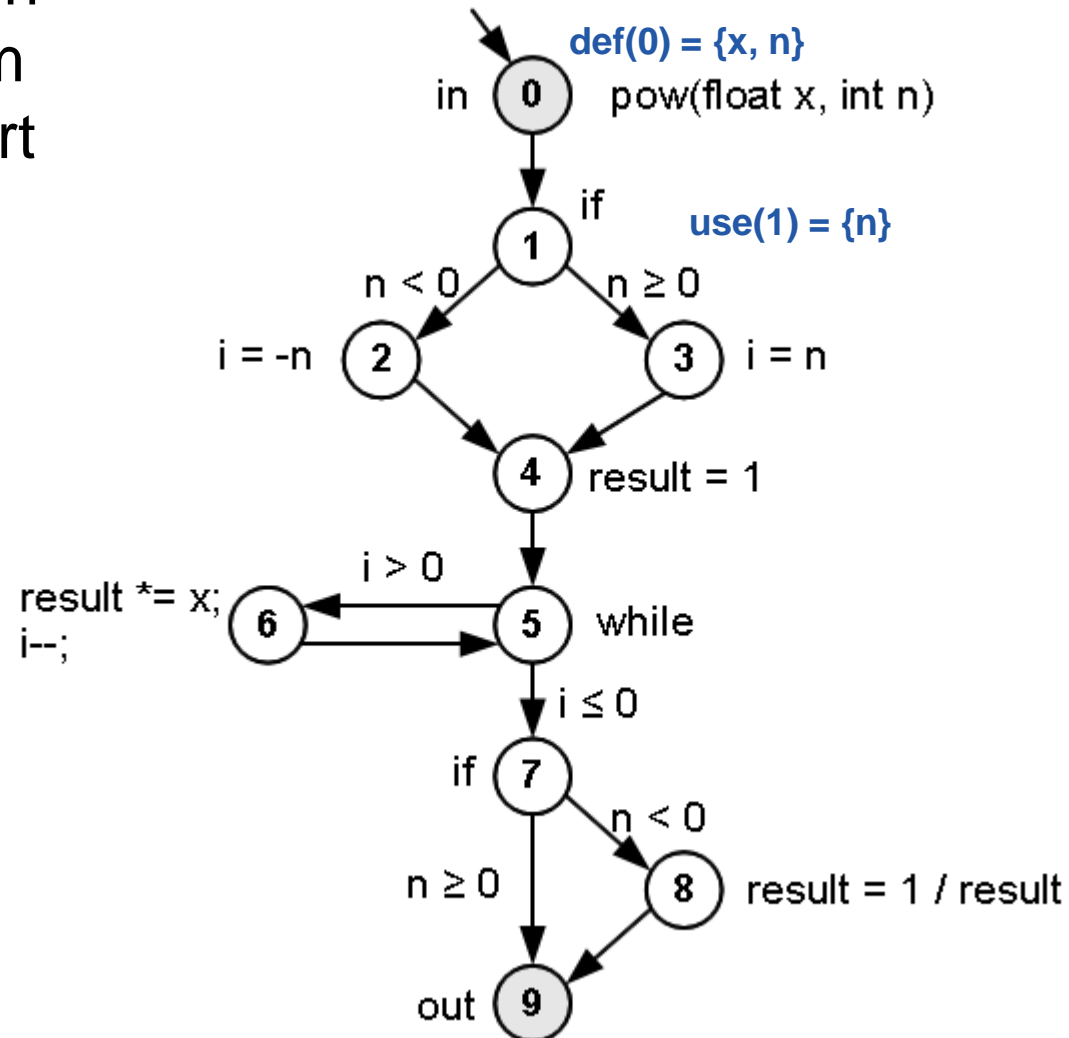
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



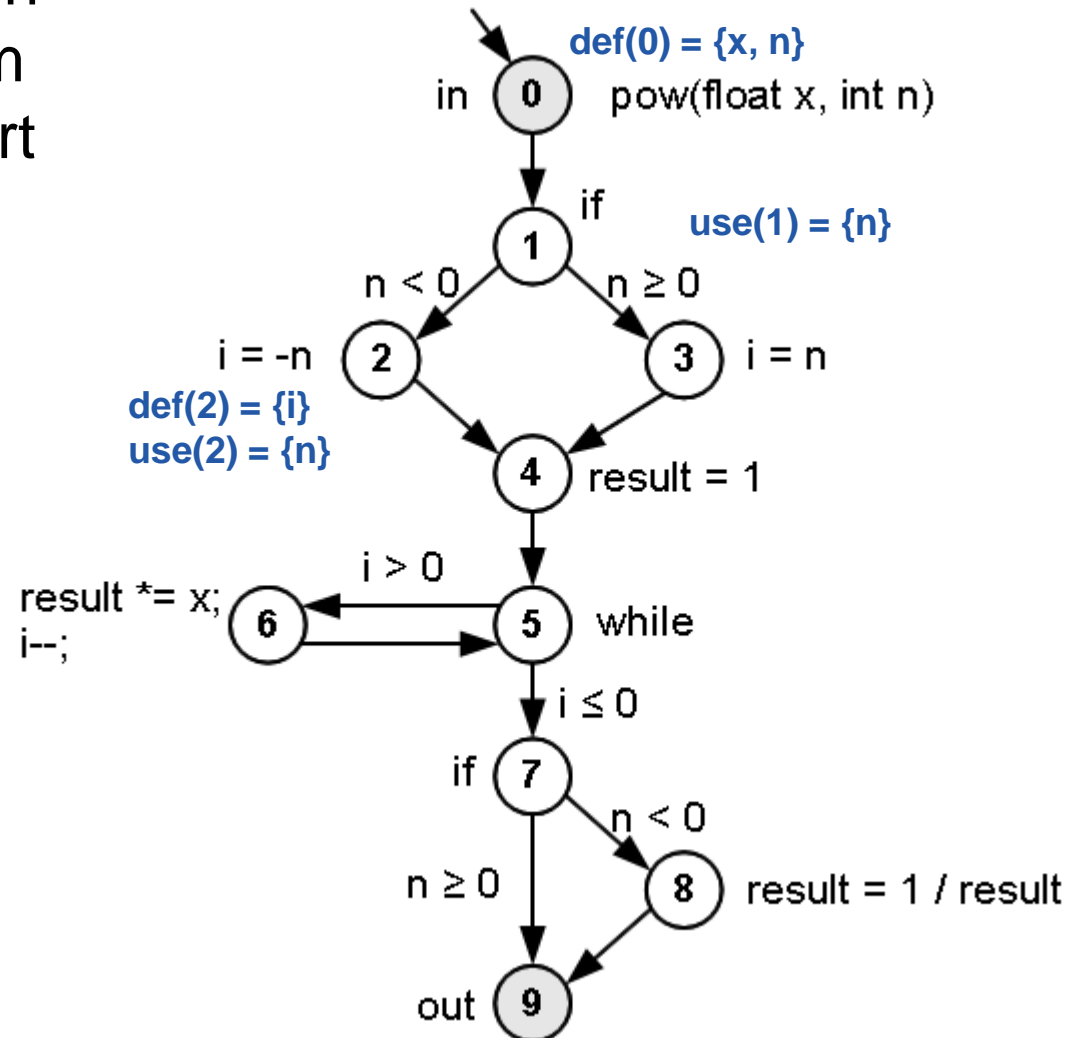
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

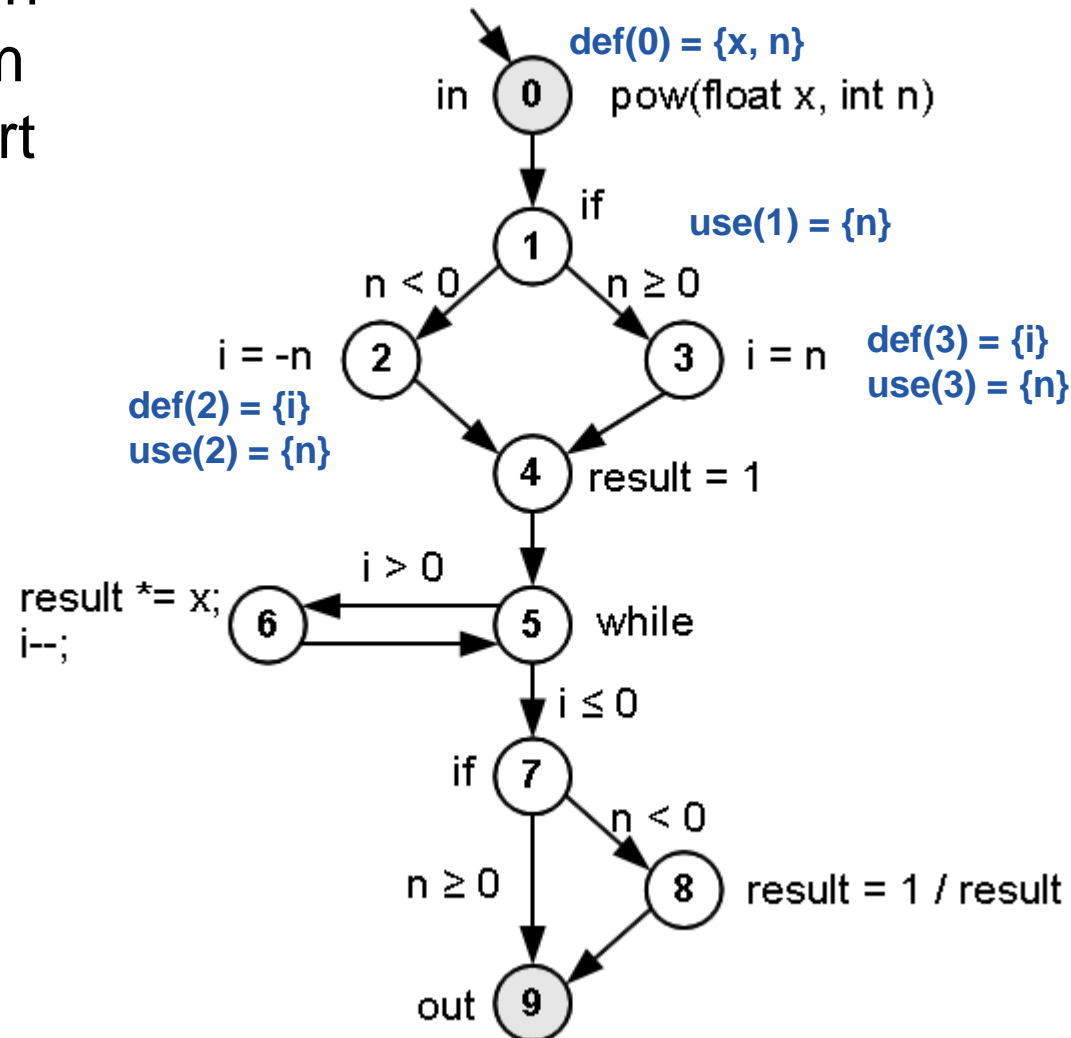
- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen





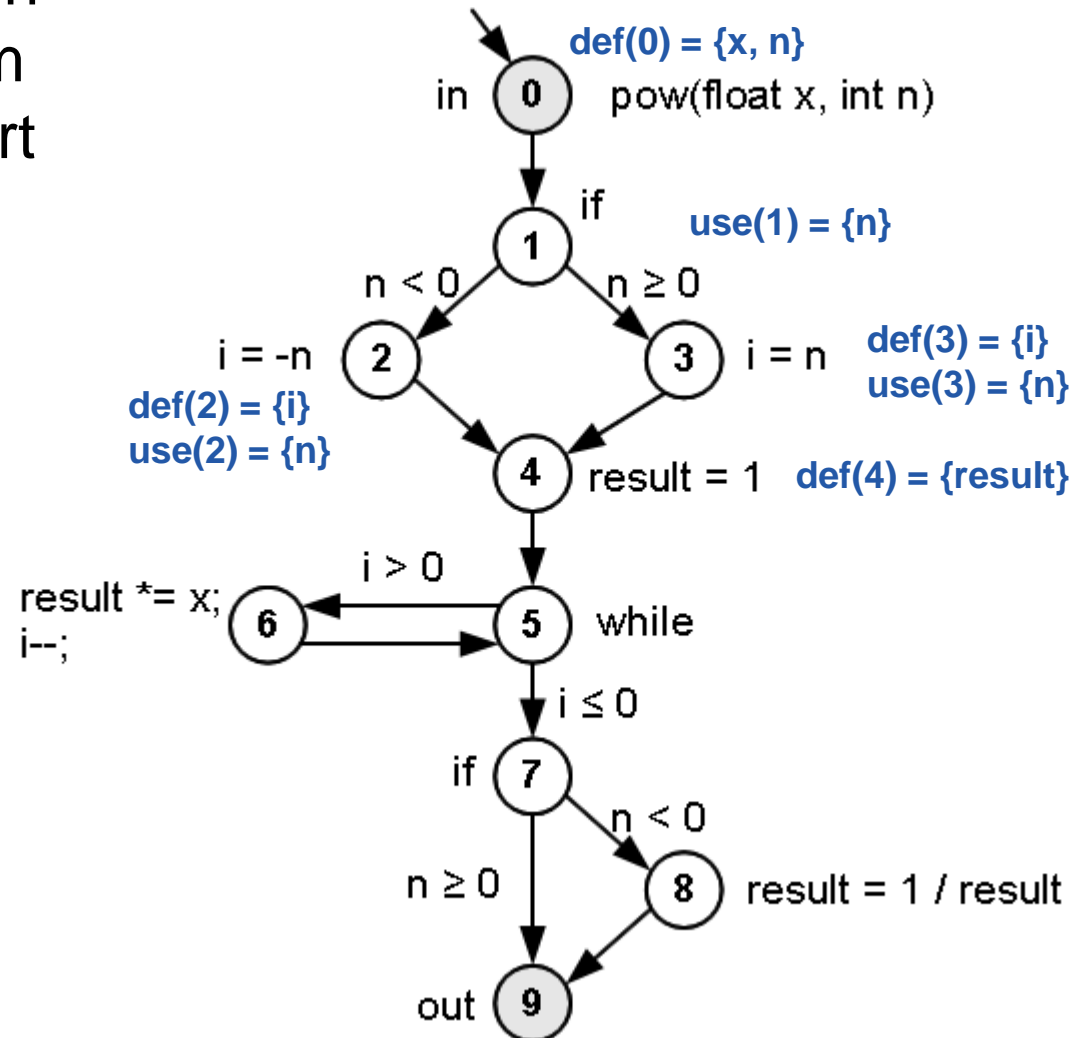
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



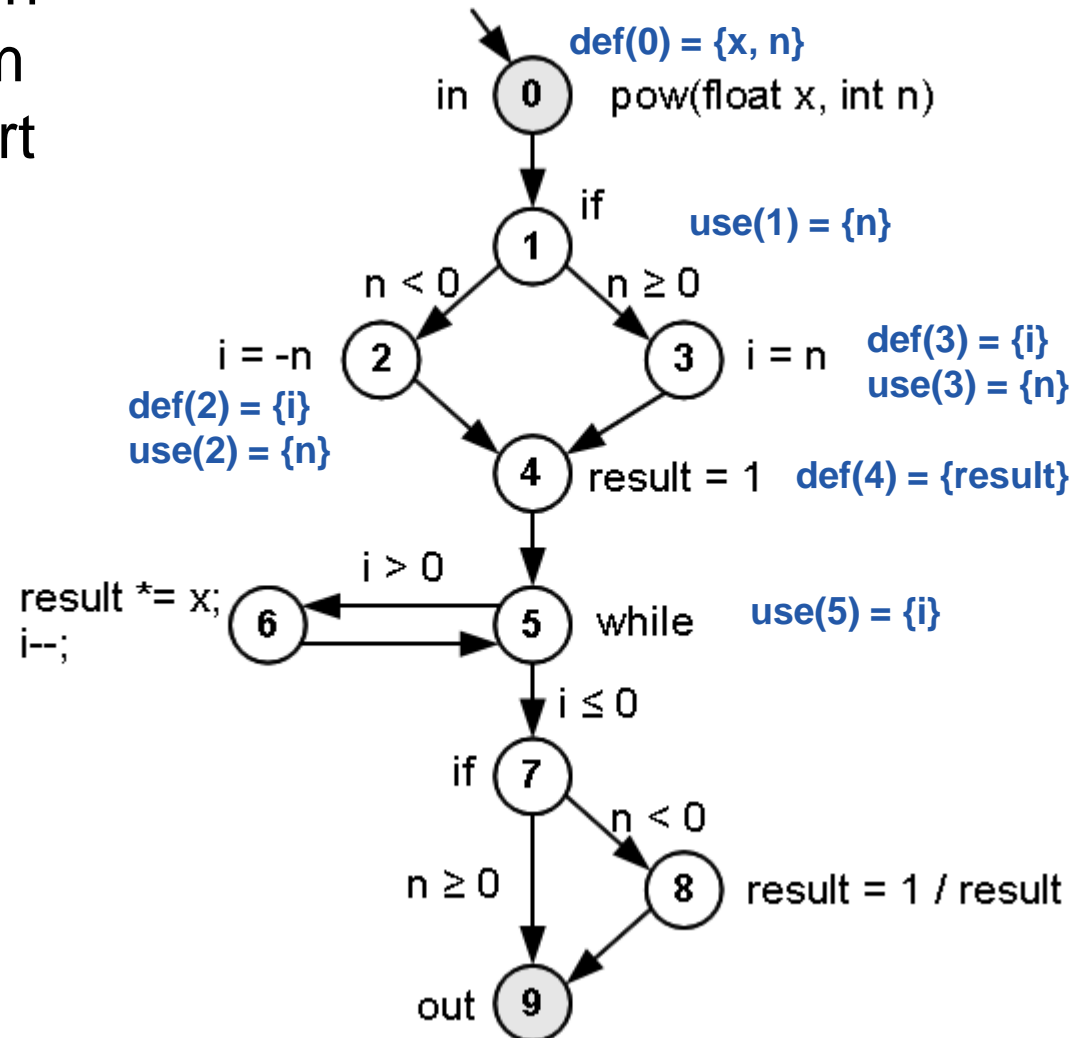
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



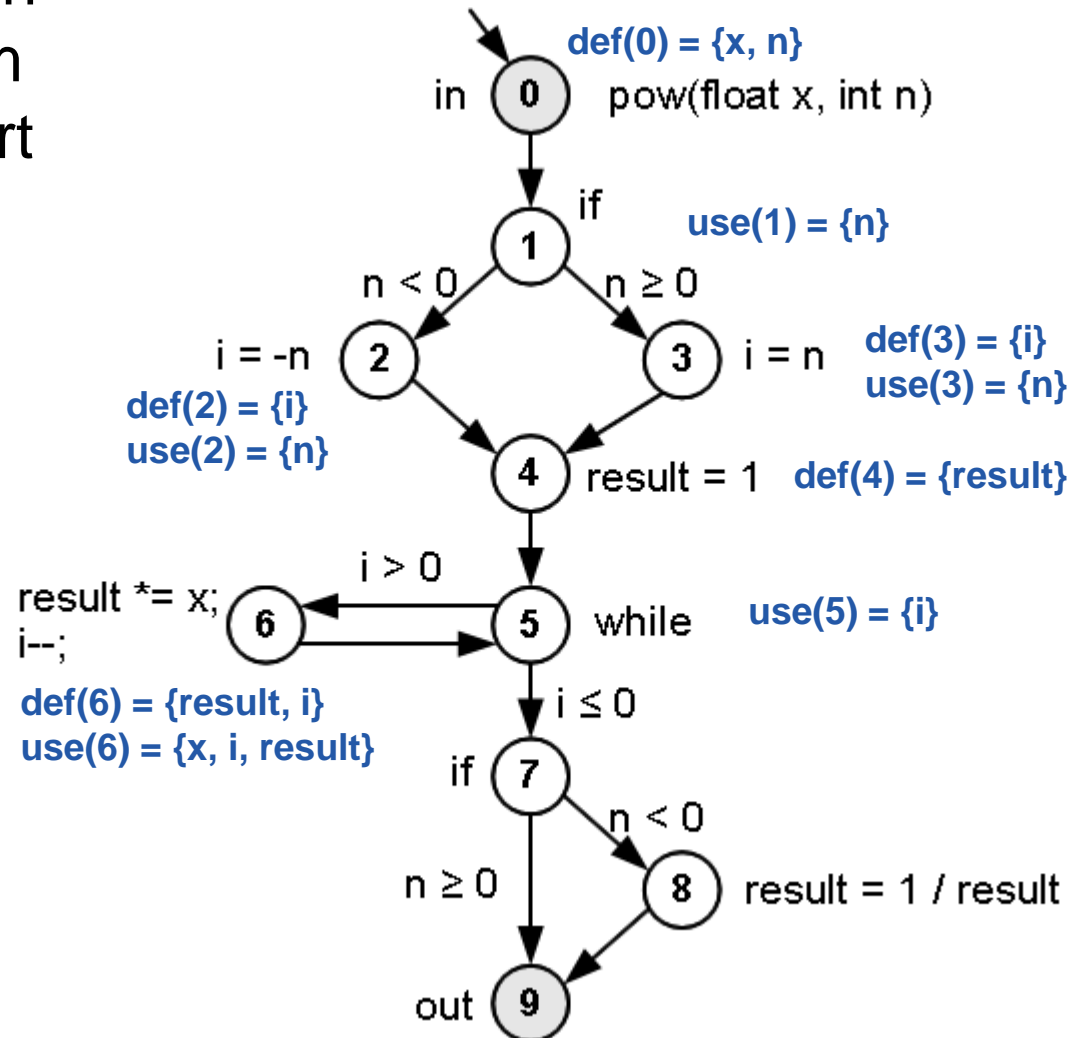
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



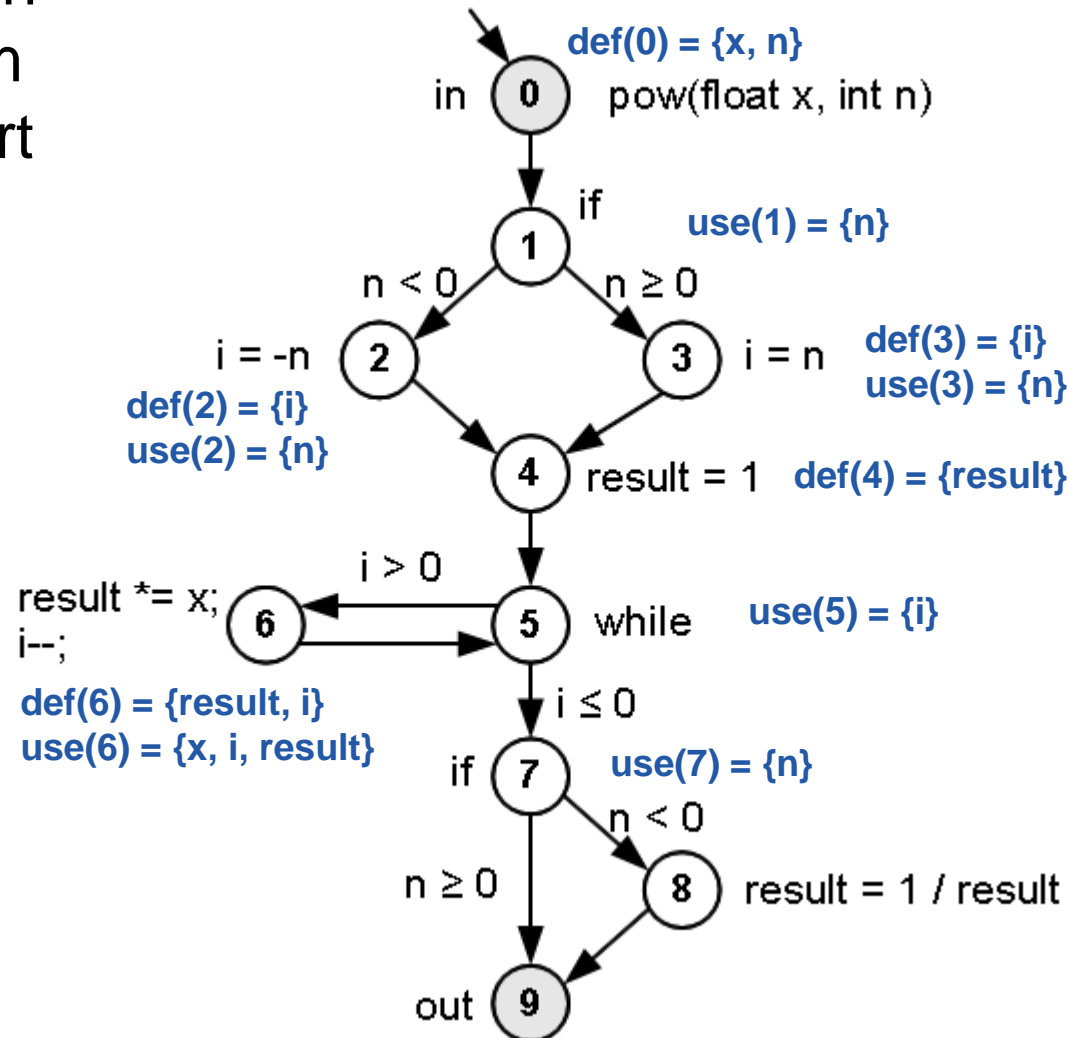
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



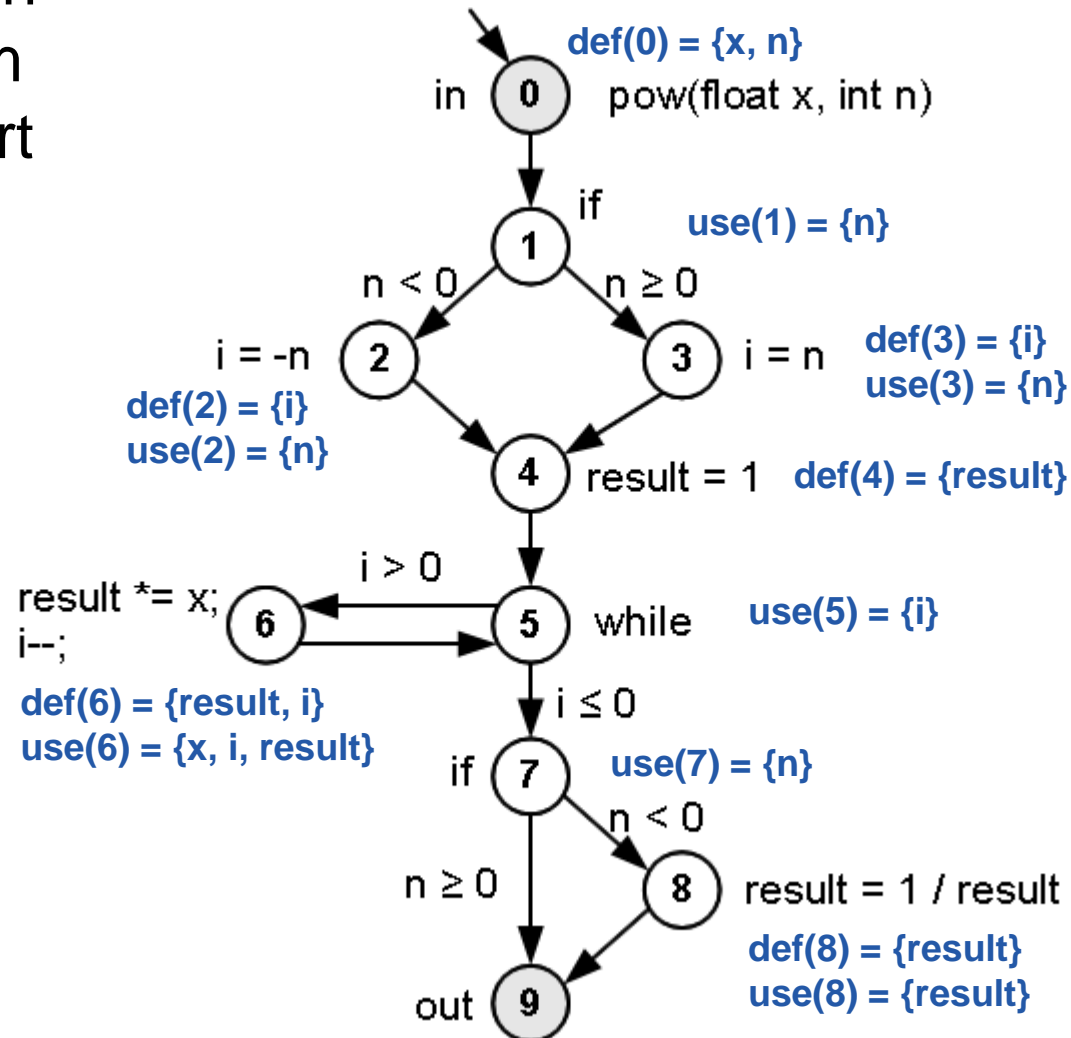
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



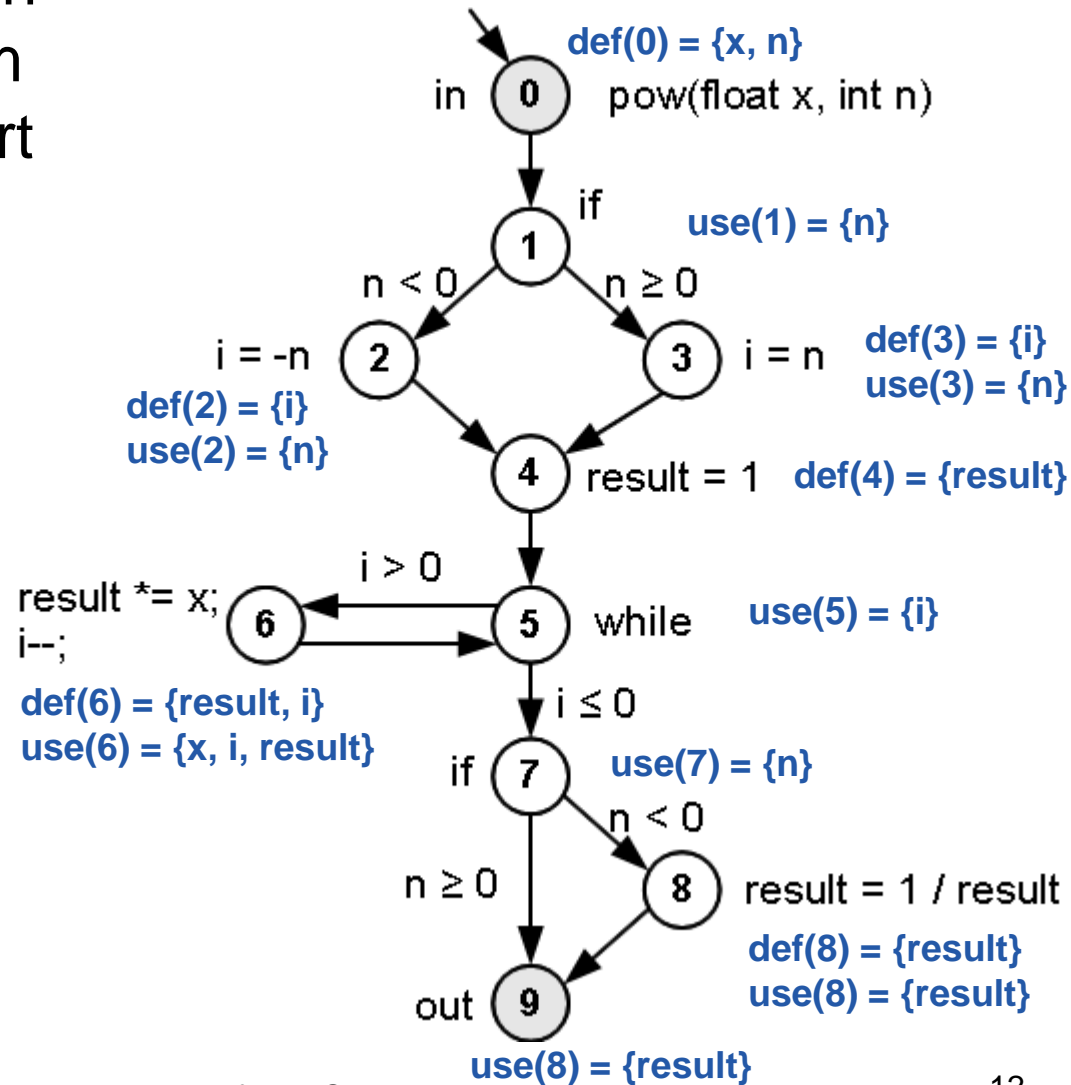
- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen



- Ein Datenflussgraph ist ein Kontrollflussgraph, der um folgende Mengen erweitert wird:

- **def(n)**: Die in Knoten n definierten Variablen
- **use(n)**: die in Knoten n gelesenen Variablen





# Defs-Uses-Überdeckungskriterien

---

- Wir benötigen folgende Definitionen:





# Defs-Uses-Überdeckungskriterien

---

- Wir benötigen folgende Definitionen:
  - Wenn Variable  $v$  in Knoten  $i$  definiert wird und in Knoten  $j$  gelesen wird, dann ist der **Pfad**  $(i, n_1, n_2, \dots, n_k, j)$  **definitionsfrei** bezüglich  $v$ , wenn  $v$  in  $n_1, n_2, \dots, n_k$  nicht definiert wird

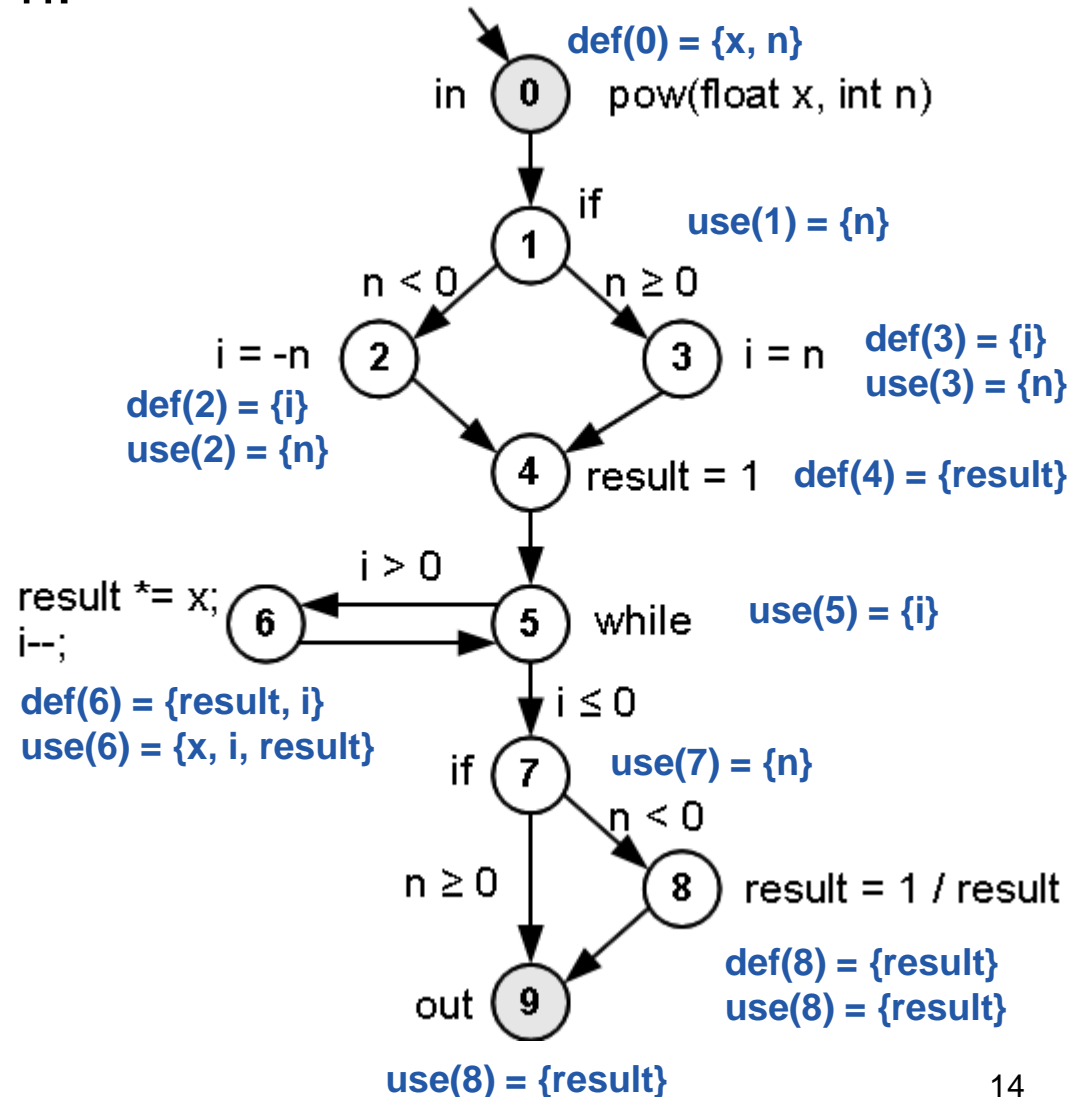
- Wir benötigen folgende Definitionen:
  - Wenn Variable  $v$  in Knoten  $i$  definiert wird und in Knoten  $j$  gelesen wird, dann ist der **Pfad**  $(i, n_1, n_2, \dots, n_k, j)$  **definitionsfrei** bezüglich  $v$ , wenn  $v$  in  $n_1, n_2, \dots, n_k$  nicht definiert wird
  - Ein **defs-use-Pfad (DU-Pfad)** für eine Variable  $v$  ist ein einfacher Pfad (ohne Zyklus, s. Def. V11) von einer Definition von  $v$  zu einer Nutzung von  $v$ , der bezüglich  $v$  definitionsfrei ist

- Wir benötigen folgende Definitionen:
  - Wenn Variable  $v$  in Knoten  $i$  definiert wird und in Knoten  $j$  gelesen wird, dann ist der **Pfad**  $(i, n_1, n_2, \dots, n_k, j)$  **definitionsfrei** bezüglich  $v$ , wenn  $v$  in  $n_1, n_2, \dots, n_k$  nicht definiert wird
  - Ein **defs-use-Pfad (DU-Pfad)** für eine Variable  $v$  ist ein einfacher Pfad (ohne Zyklus, s. Def. V11) von einer Definition von  $v$  zu einer Nutzung von  $v$ , der bezüglich  $v$  definitionsfrei ist
  - Die **Pfadmenge**  $du(n, n', v)$  ist die Menge aller DU-Pfade von  $v$ , die in  $n$  beginnen und in  $n'$  enden

- Wir benötigen folgende Definitionen:
  - Wenn Variable  $v$  in Knoten  $i$  definiert wird und in Knoten  $j$  gelesen wird, dann ist der **Pfad**  $(i, n_1, n_2, \dots, n_k, j)$  **definitionsfrei** bezüglich  $v$ , wenn  $v$  in  $n_1, n_2, \dots, n_k$  nicht definiert wird
  - Ein **defs-use-Pfad (DU-Pfad)** für eine Variable  $v$  ist ein einfacher Pfad (ohne Zyklus, s. Def. V11) von einer Definition von  $v$  zu einer Nutzung von  $v$ , der bezüglich  $v$  definitionsfrei ist
  - Die **Pfadmenge**  $du(n, n', v)$  ist die Menge aller DU-Pfade von  $v$ , die in  $n$  beginnen und in  $n'$  enden
  - Die **Pfadmenge**  $du(n, v)$  ist die Menge aller DU-Pfade von  $v$ , die in  $n$  beginnen

- Wir benötigen folgende Definitionen:
  - Wenn Variable  $v$  in Knoten  $i$  definiert wird und in Knoten  $j$  gelesen wird, dann ist der **Pfad**  $(i, n_1, n_2, \dots, n_k, j)$  **definitionsfrei** bezüglich  $v$ , wenn  $v$  in  $n_1, n_2, \dots, n_k$  nicht definiert wird
  - Ein **defs-use-Pfad (DU-Pfad)** für eine Variable  $v$  ist ein einfacher Pfad (ohne Zyklus, s. Def. V11) von einer Definition von  $v$  zu einer Nutzung von  $v$ , der bezüglich  $v$  definitionsfrei ist
  - Die **Pfadmenge**  $du(n, n', v)$  ist die Menge aller DU-Pfade von  $v$ , die in  $n$  beginnen und in  $n'$  enden
  - Die **Pfadmenge**  $du(n, v)$  ist die Menge aller DU-Pfade von  $v$ , die in  $n$  beginnen
    - $du(n, v)$  ist gleich der Vereinigung der  $du(n, n', v)$  für alle  $n'$ :  
also  $du(n, v) = \bigcup_{n'} du(n, n', v)$

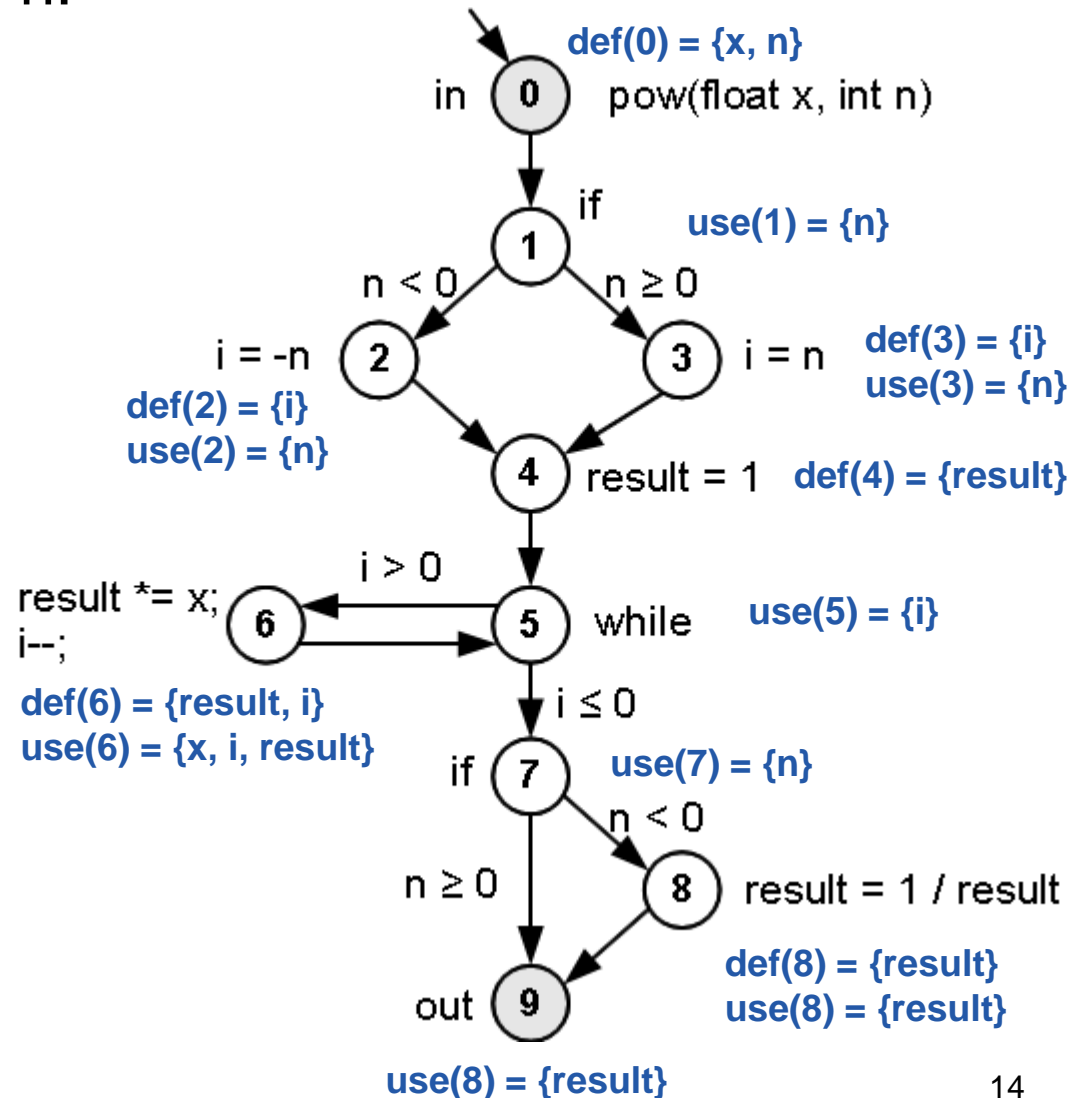
- Ein **DU-Pfad** der Variable  $n$ :





# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

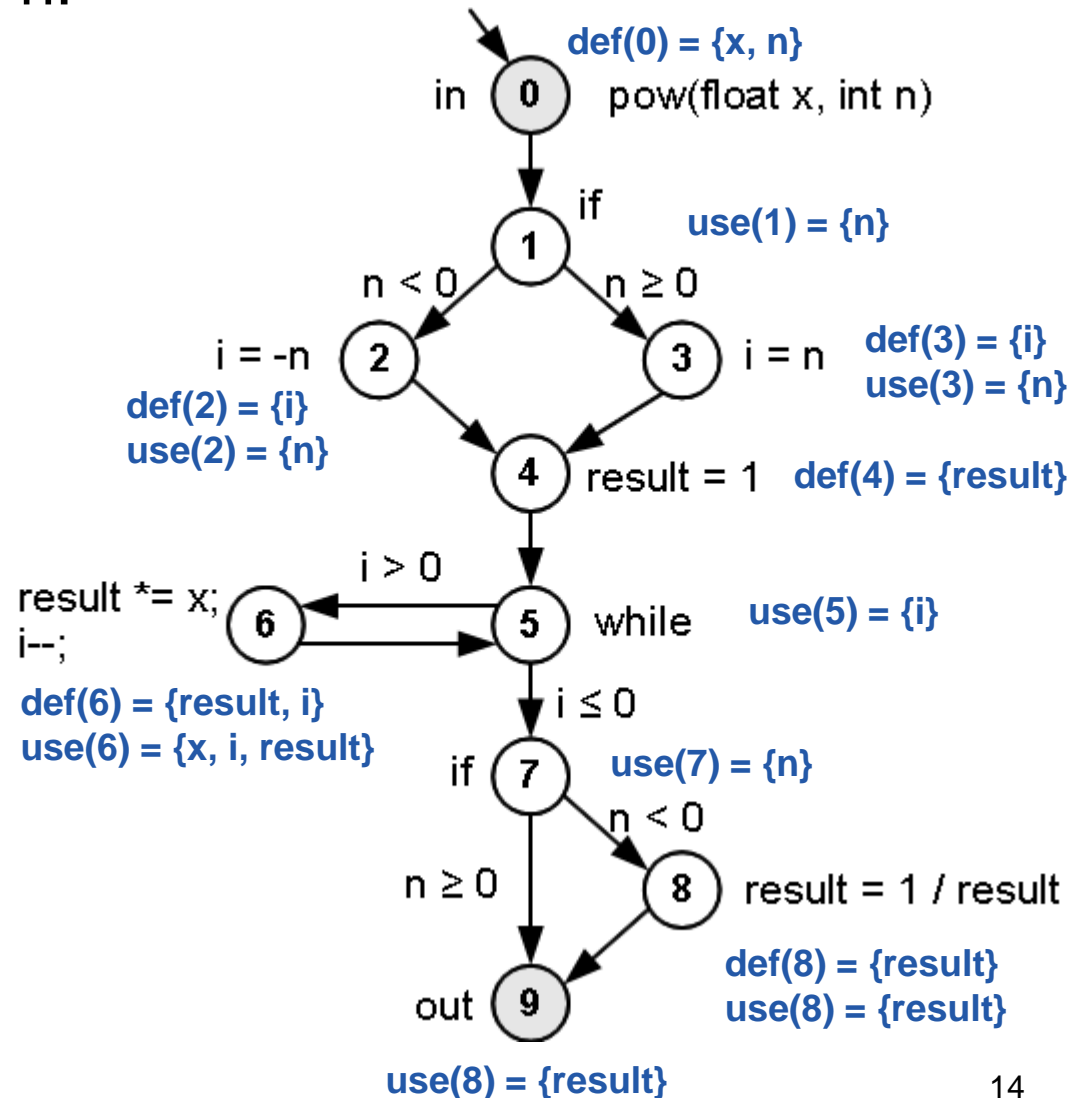
- Ein **DU-Pfad** der Variable  $n$ :
  - Z.B.  $(0, 1)$





# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :
  - Z.B.  $(0, 1)$
- **$du(4, 9, \text{result})$**





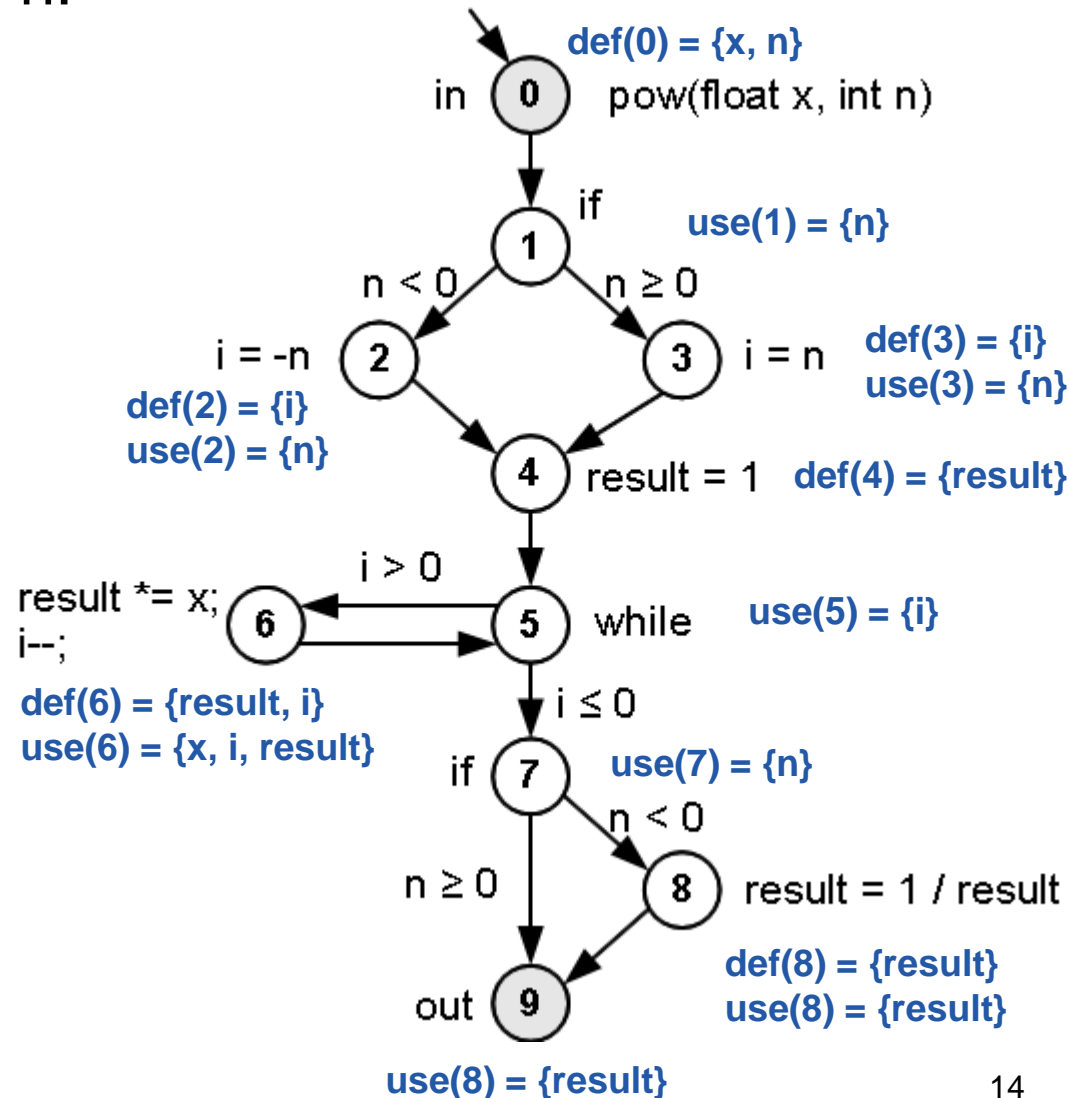


# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$





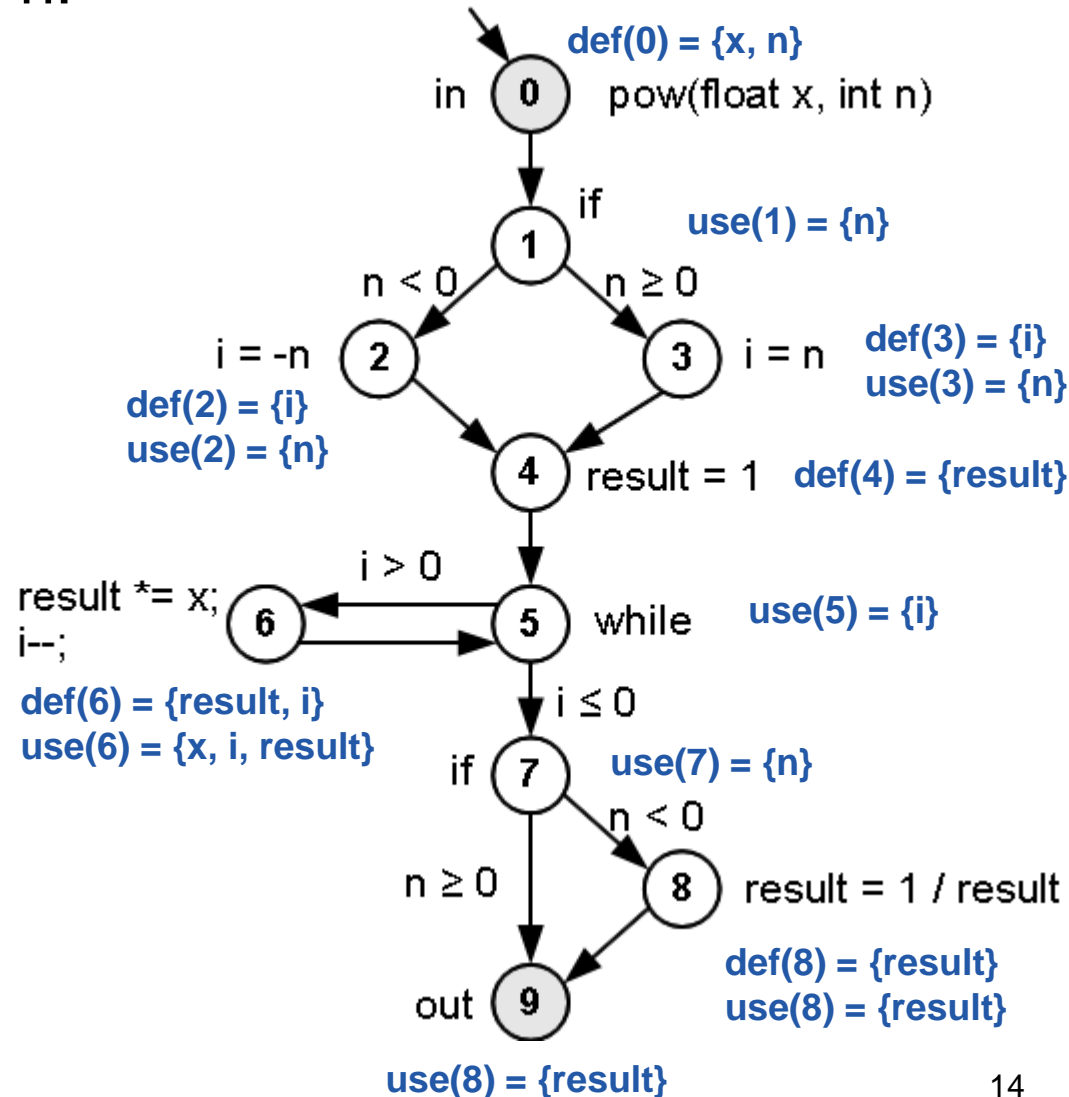
# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$**





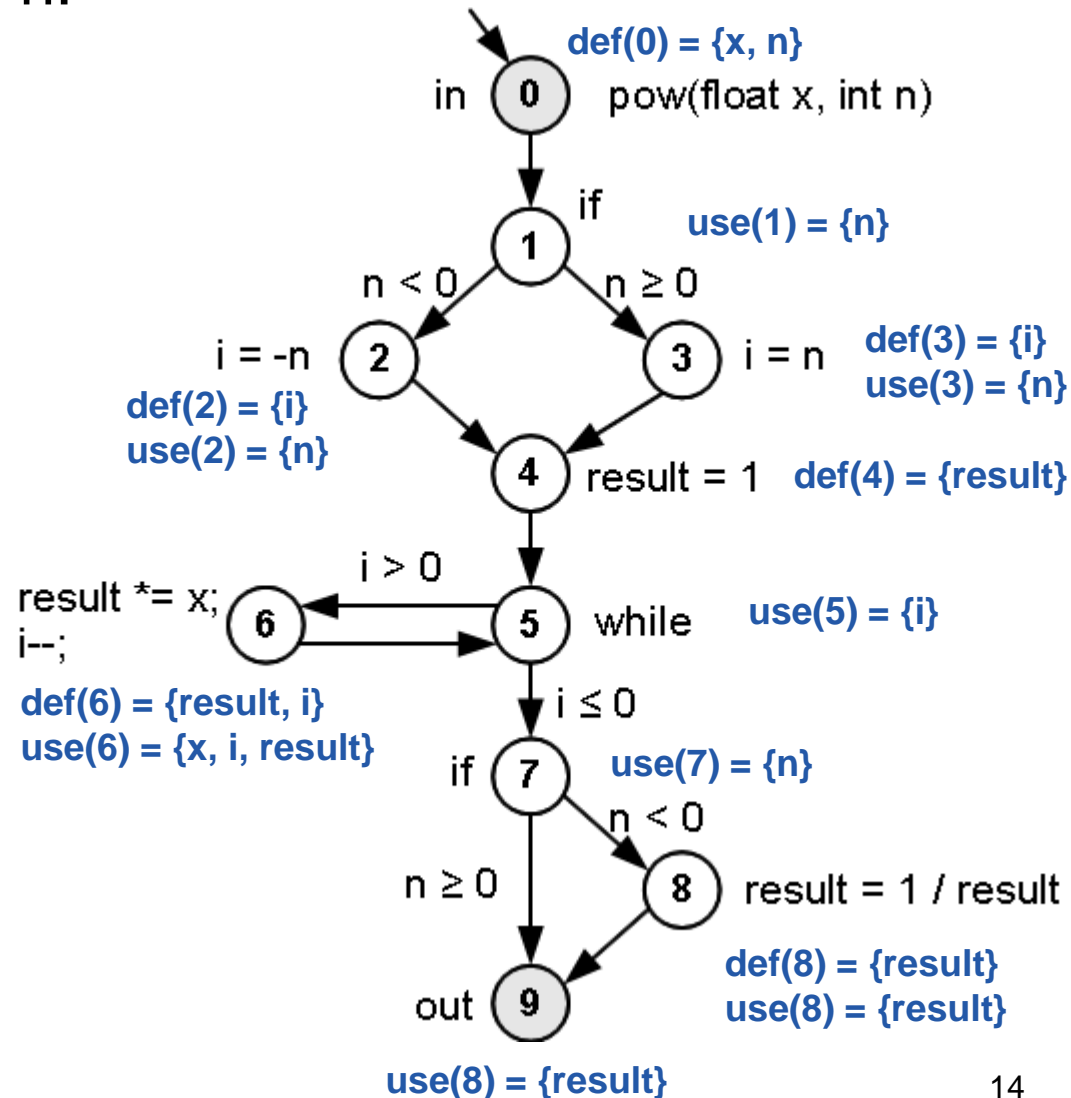
# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$**   
 $= \{(4, 5, 7, 9), (4, 5, 6)\}$





# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

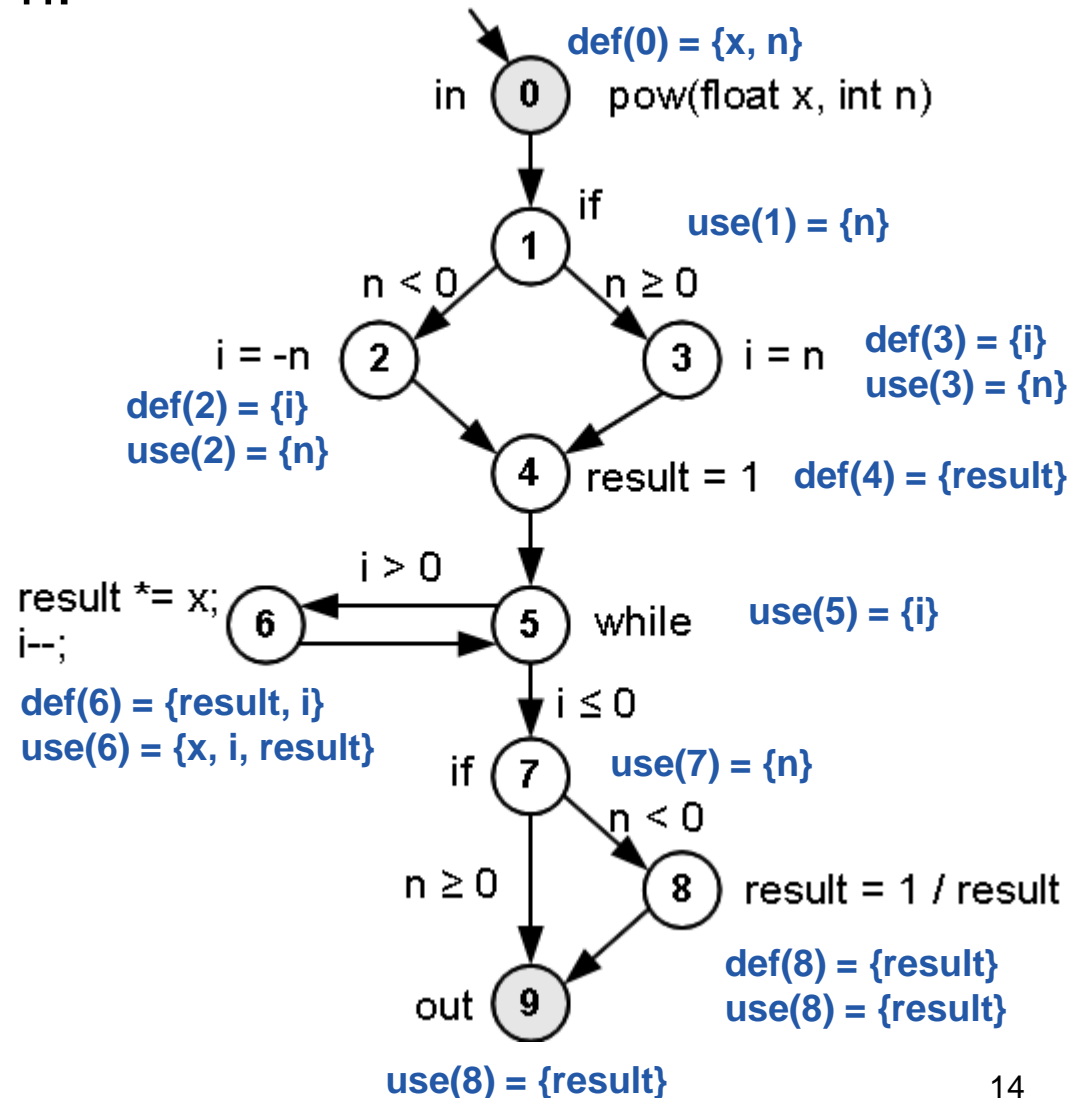
- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$**   
 $= \{(4, 5, 7, 9), (4, 5, 6)\}$

**Frage:** Wieso nicht  
 $(4, 5, 7, 8)$ ?



- Ein **DU-Pfad** der Variable  $n$ :

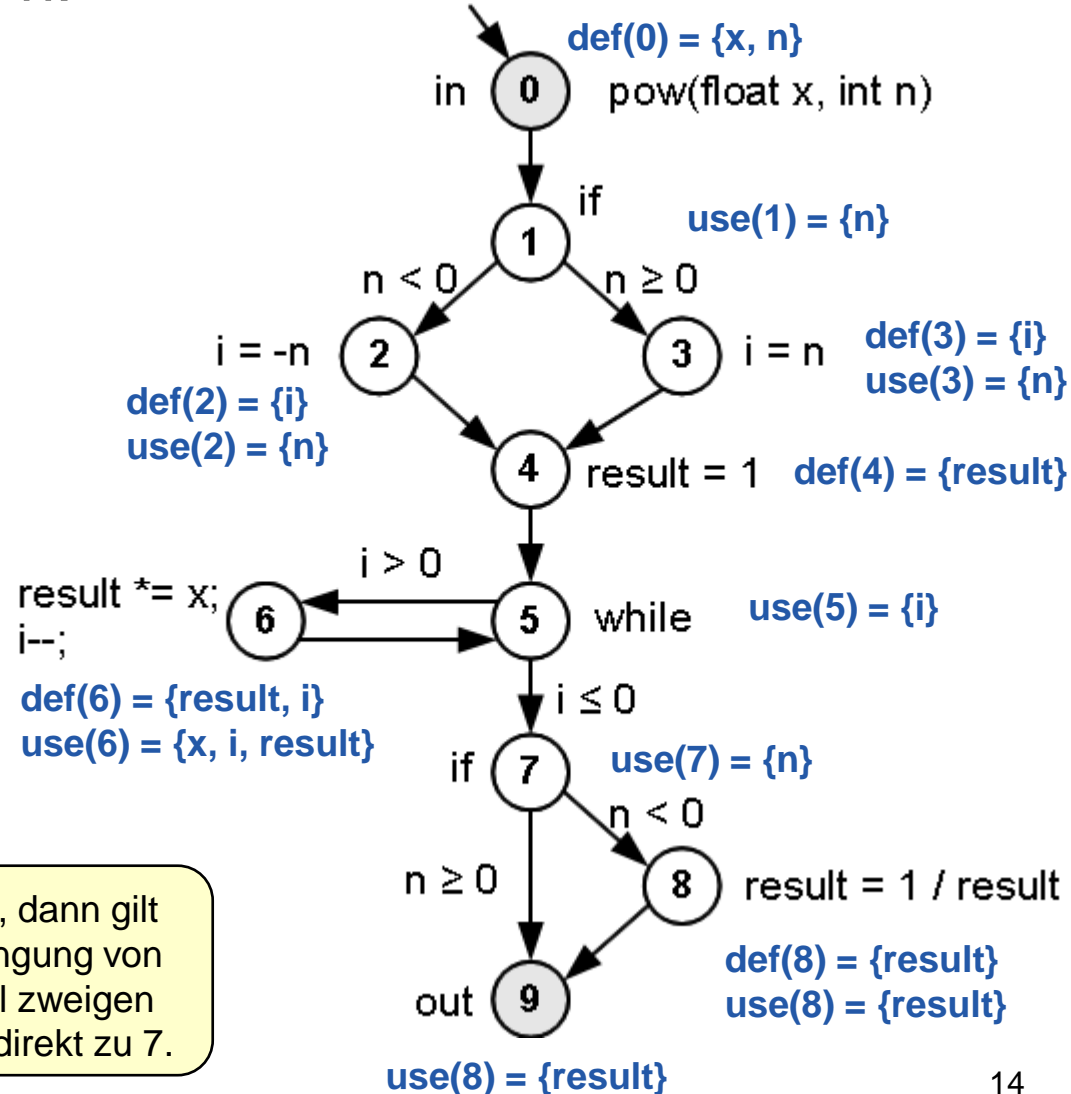
– Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$**   
 $= \{(4, 5, 7, 9), (4, 5, 6)\}$

**Frage:** Wieso nicht  $(4, 5, 7, 8)$ ?

**Antwort:** Wenn wir von 7 nach 8 gehen, dann gilt  $n < 0$ . Dann gilt aber auch  $i = -n$  (if-Bedingung von Knoten 1) und auch  $i > 1$ . In diesem Fall zweigen wir von 5 also zu 6 ab und können nicht direkt zu 7.





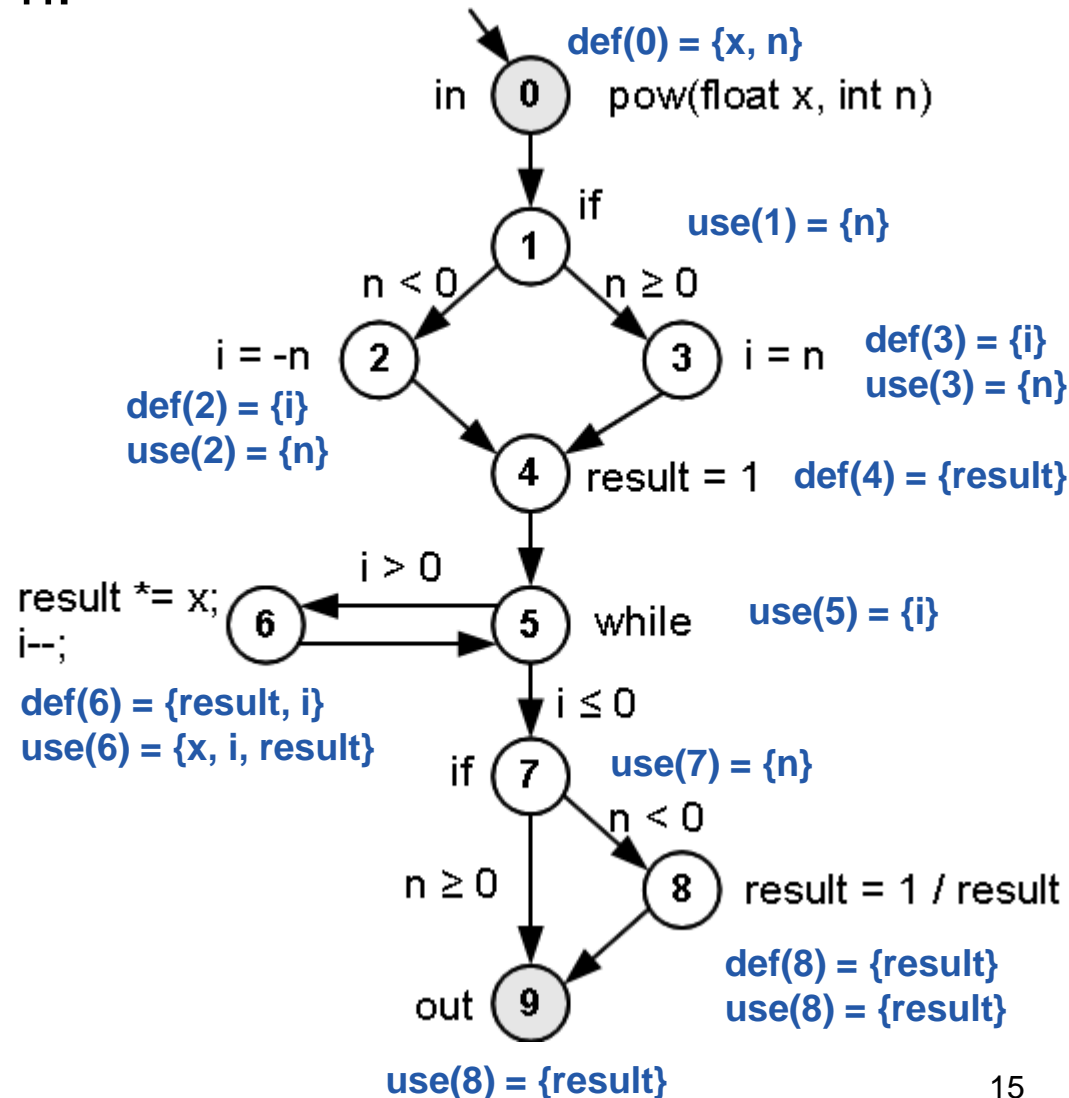
# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$**   
 $= \{(4, 5, 7, 9), (4, 5, 6)\}$



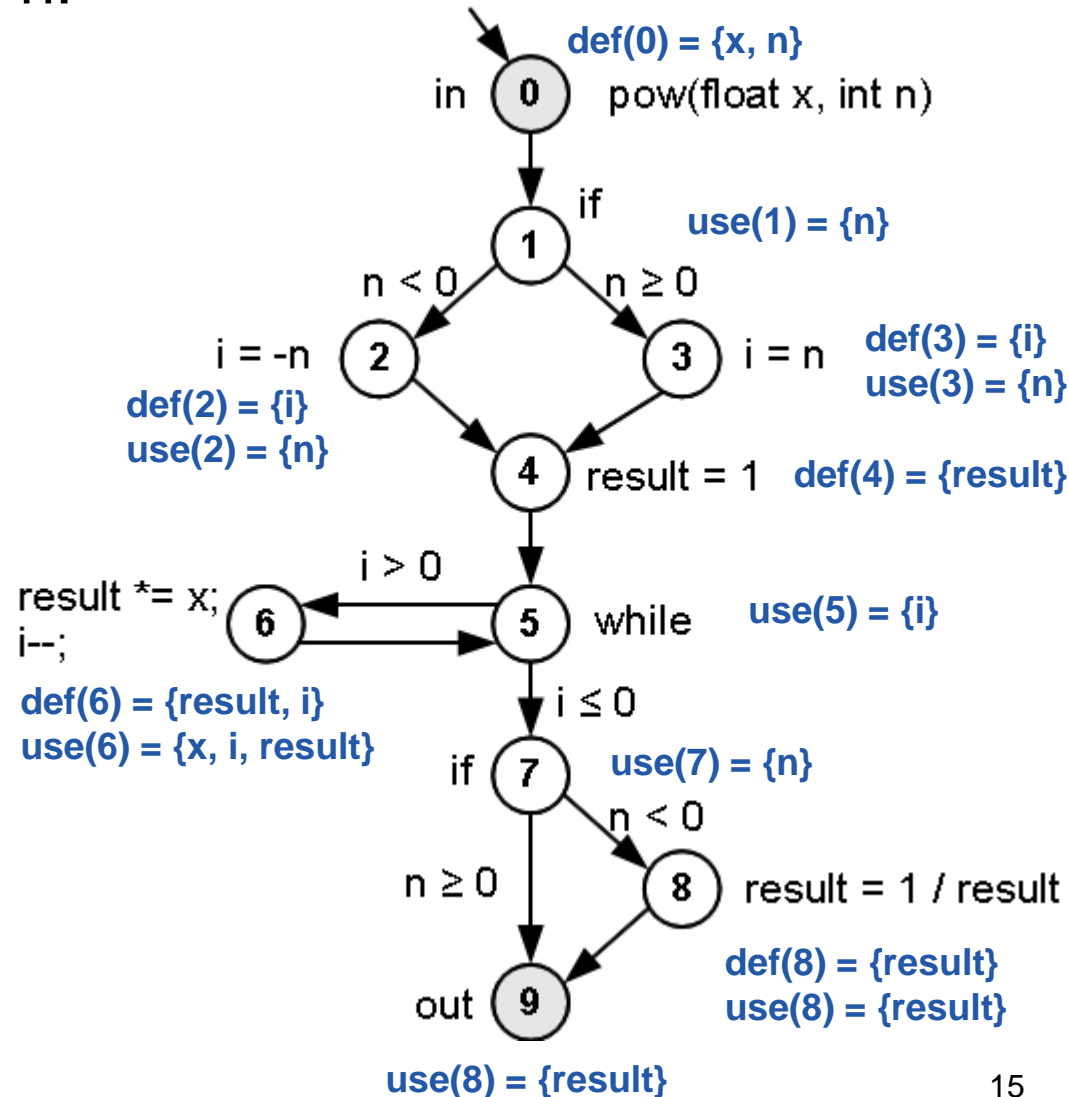
- Ein **DU-Pfad** der Variable  $n$ :

– Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$   
 $= \{(4, 5, 7, 9)\}$

- $du(4, \text{result})$   
 $= \{(4, 5, 7, 9), (4, 5, 6)\}$

- Gibt es im Bsp. eine Menge  $du(n, n', v)$  mit mehr als einem Element?





# DU-Pfad, Pfadmengen $du(n, n', v)$ , $du(n, v)$

- Ein **DU-Pfad** der Variable  $n$ :

- Z.B.  $(0, 1)$

- $du(4, 9, \text{result})$**

$= \{(4, 5, 7, 9)\}$

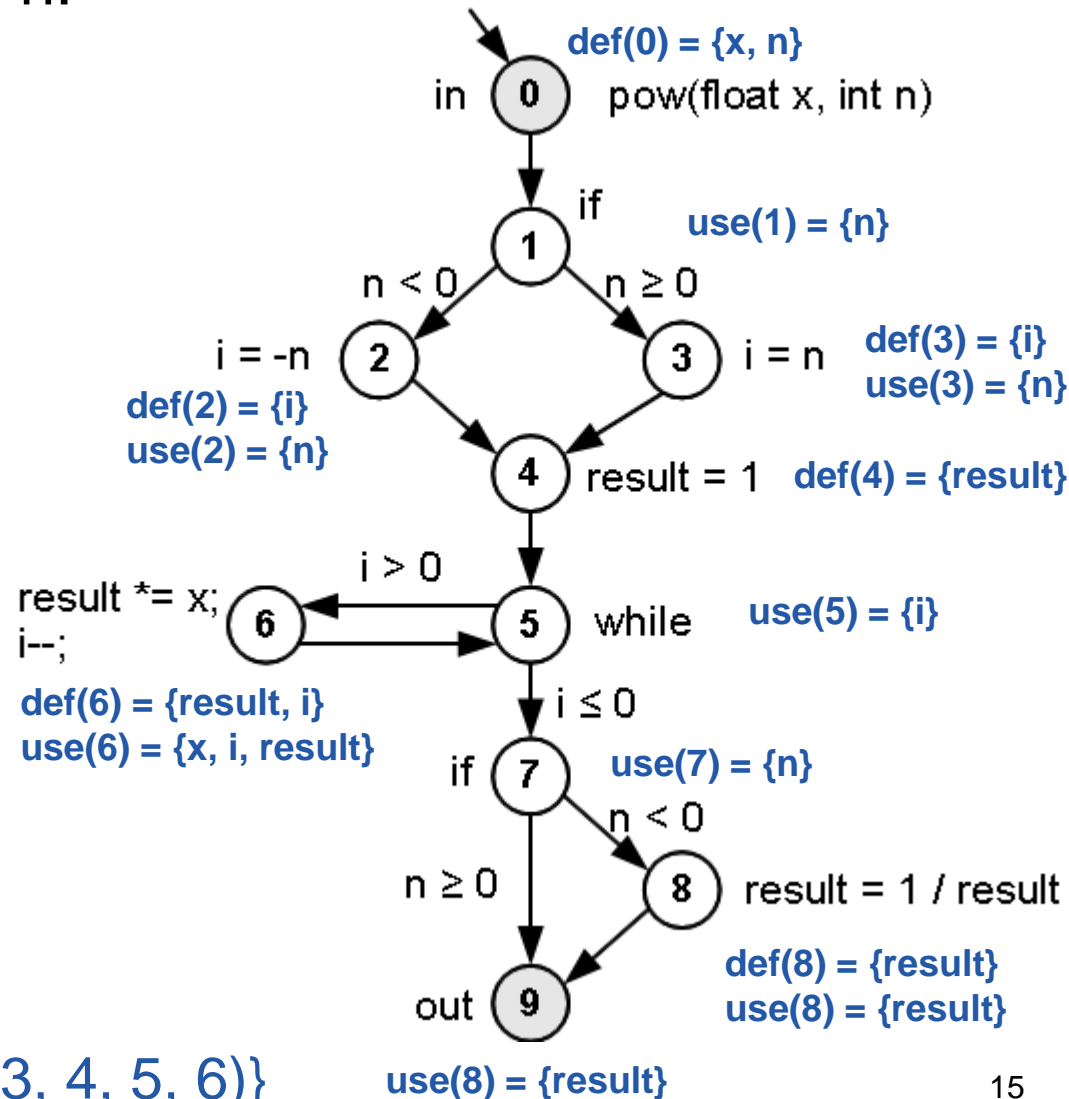
- $du(4, \text{result})$**

$= \{(4, 5, 7, 9),$   
 $(4, 5, 6)\}$

- Gibt es im Bsp. eine Menge  $du(n, n', v)$  mit mehr als einem Element?

- Ja,  **$du(0, 6, x)$**

$= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$







# Defs-Uses-Überdeckungskriterien

## All-Definitions

---

- Überdeckungskriterium: **All-Definitions**
  - **Für alle Definitionen einer Variable**, überdecke mindestens einen bezüglich dieser Variablen definitionsfreien Pfad zu einem lesenden Zugriff (c-use oder p-use) dieser Variablen



# Defs-Uses-Überdeckungskriterien

## All-Definitions

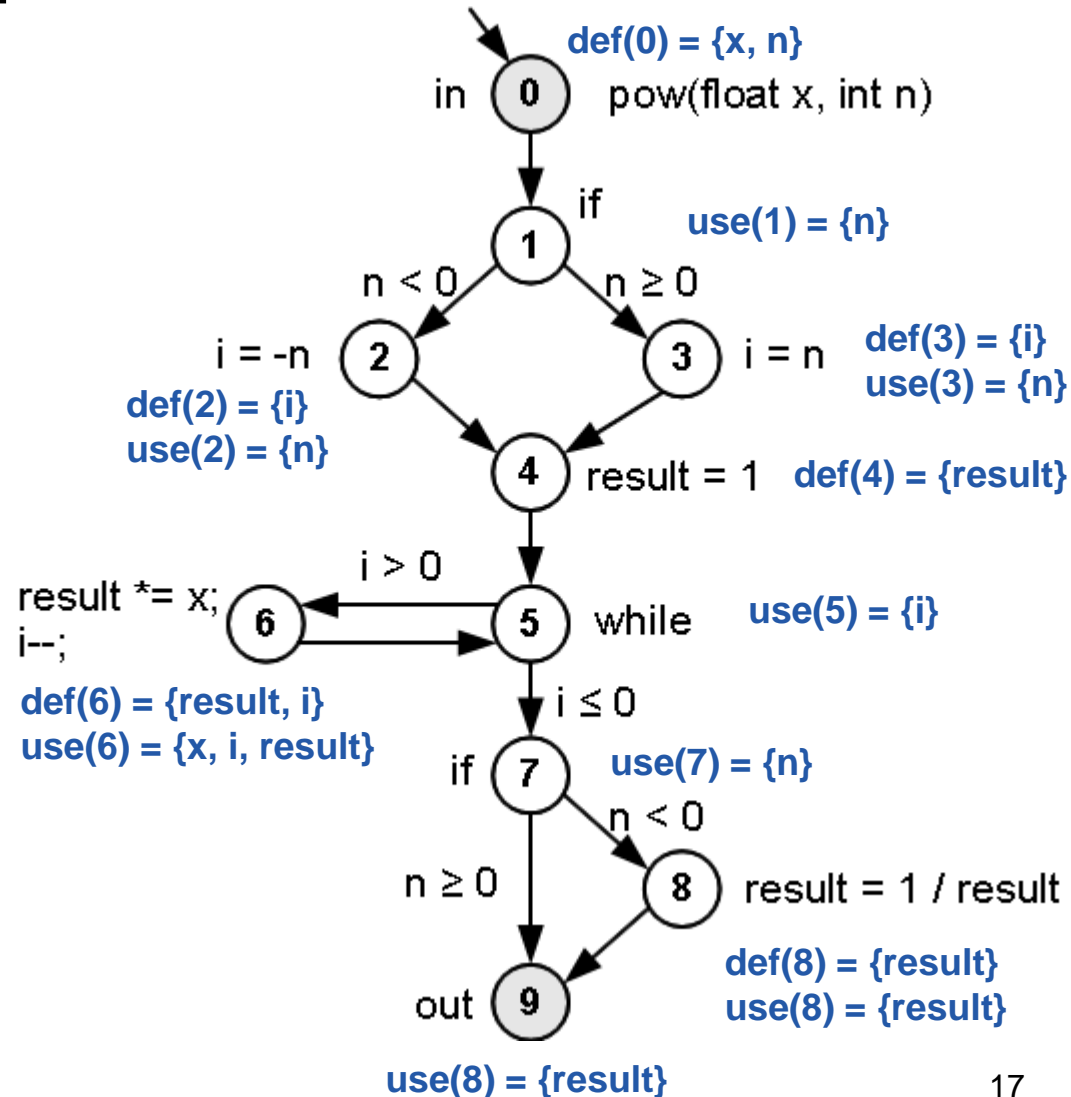
---

- Überdeckungskriterium: **All-Definitions**
  - **Für alle Definitionen einer Variable**, überdecke mindestens einen bezüglich dieser Variablen definitionsfreien Pfad zu einem lesenden Zugriff (c-use oder p-use) dieser Variablen
  - Etwas formaler: Für **jede Pfadmenge  $du(n, v)$**  muss **mindestens ein Pfad** aus dieser Menge überdeckt werden

# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

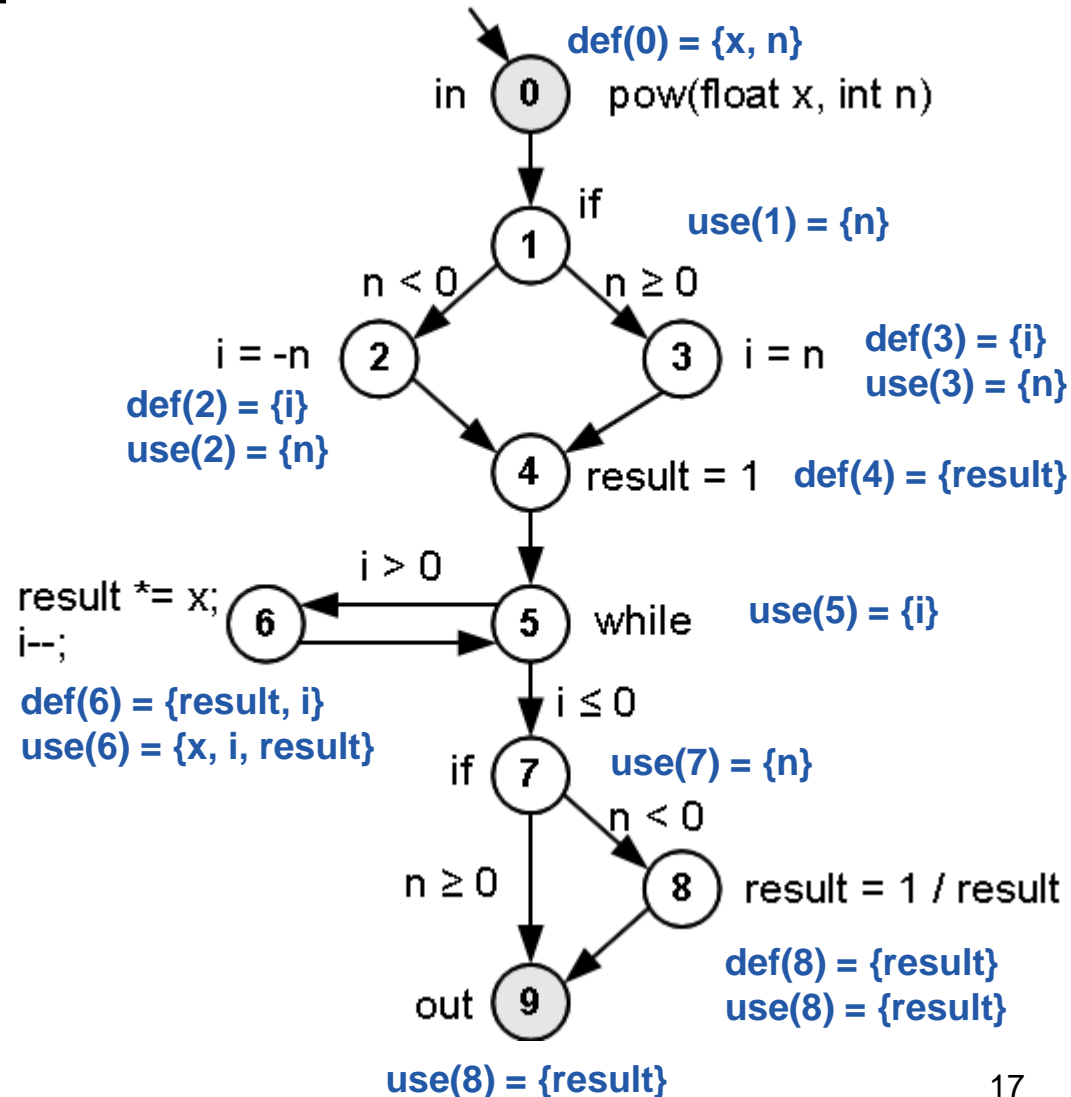


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmengen  $du(n, v)$ :

$du(0, x)$



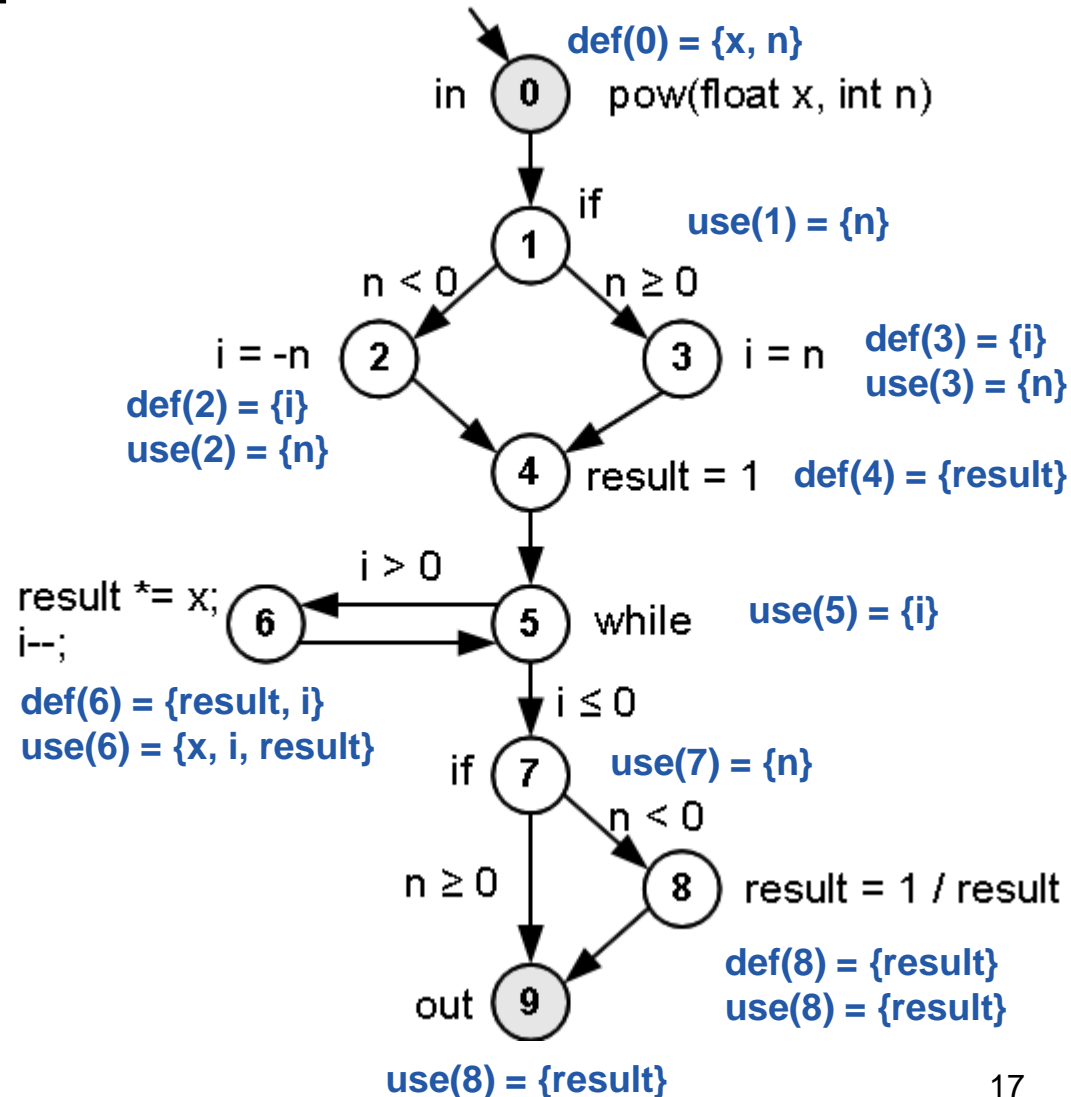


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

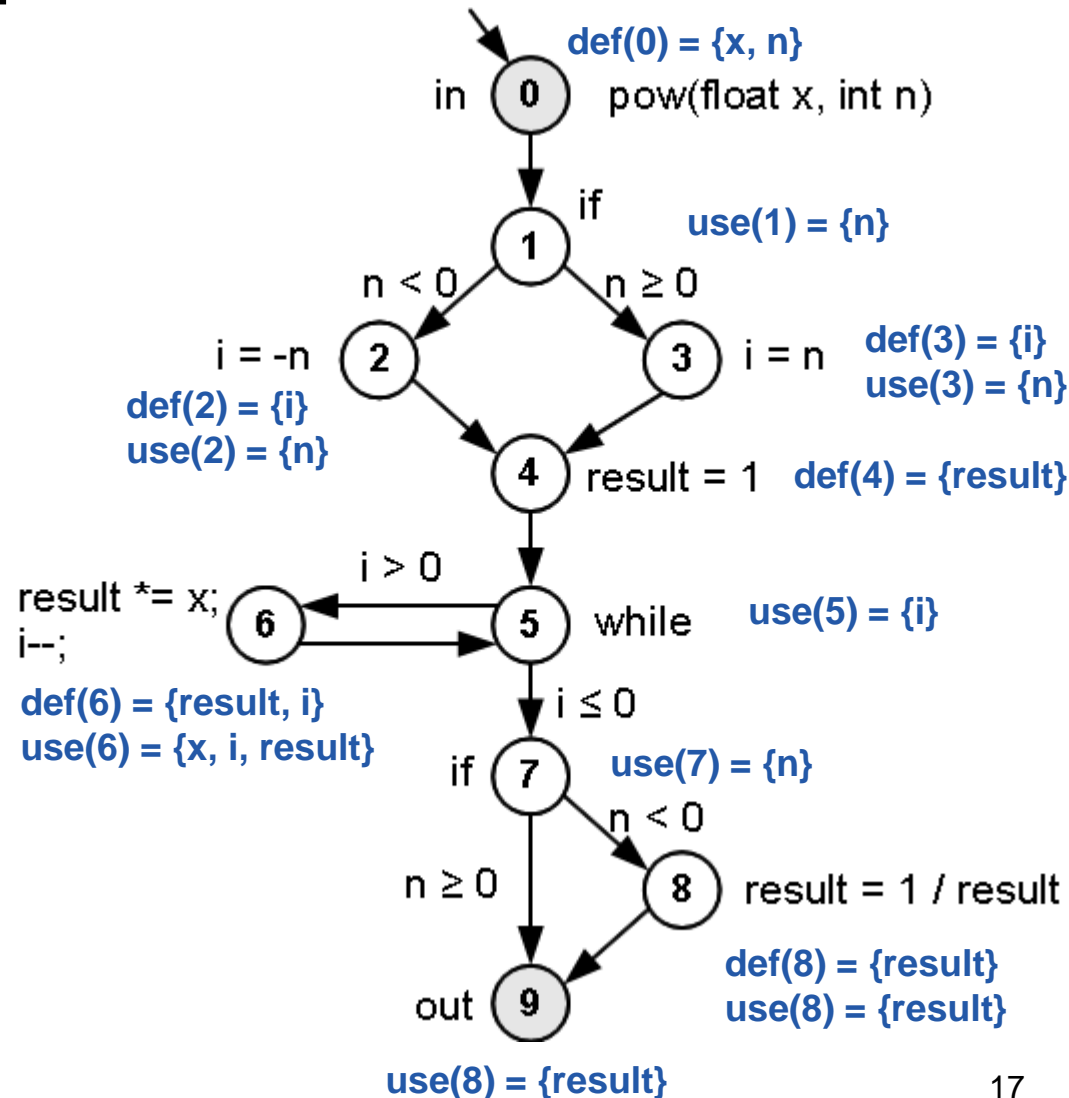


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$



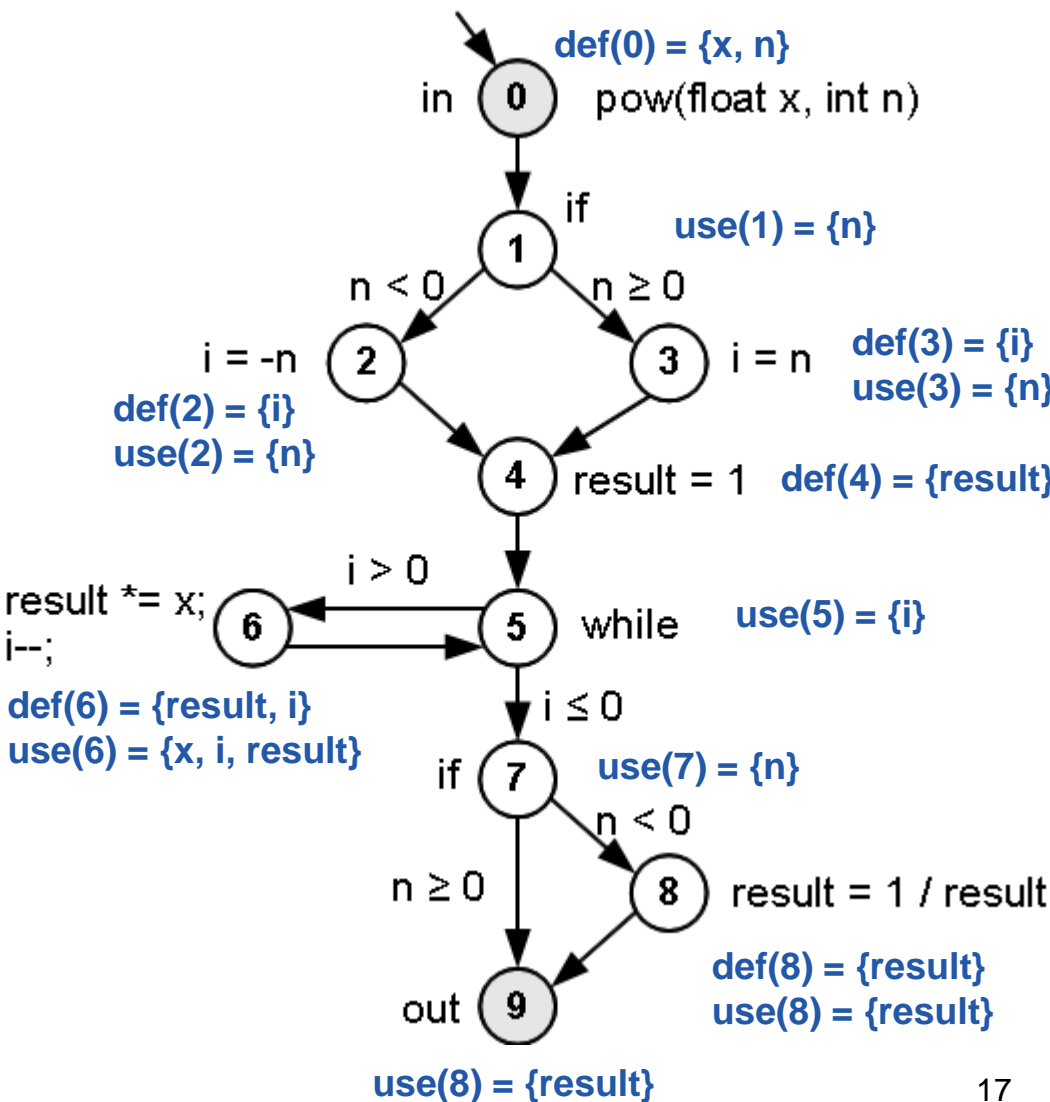


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

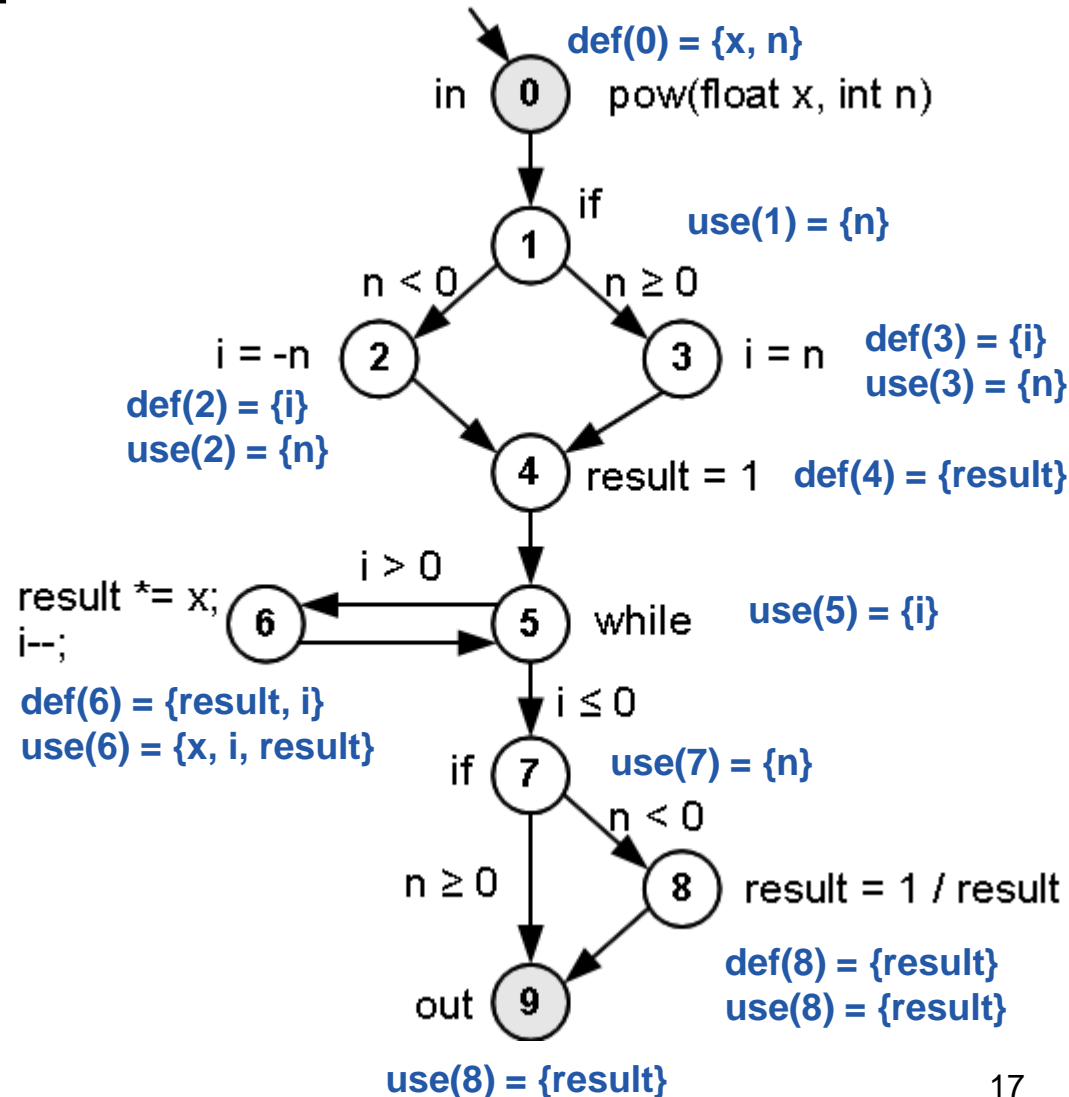


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$   
 $= \{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i)$



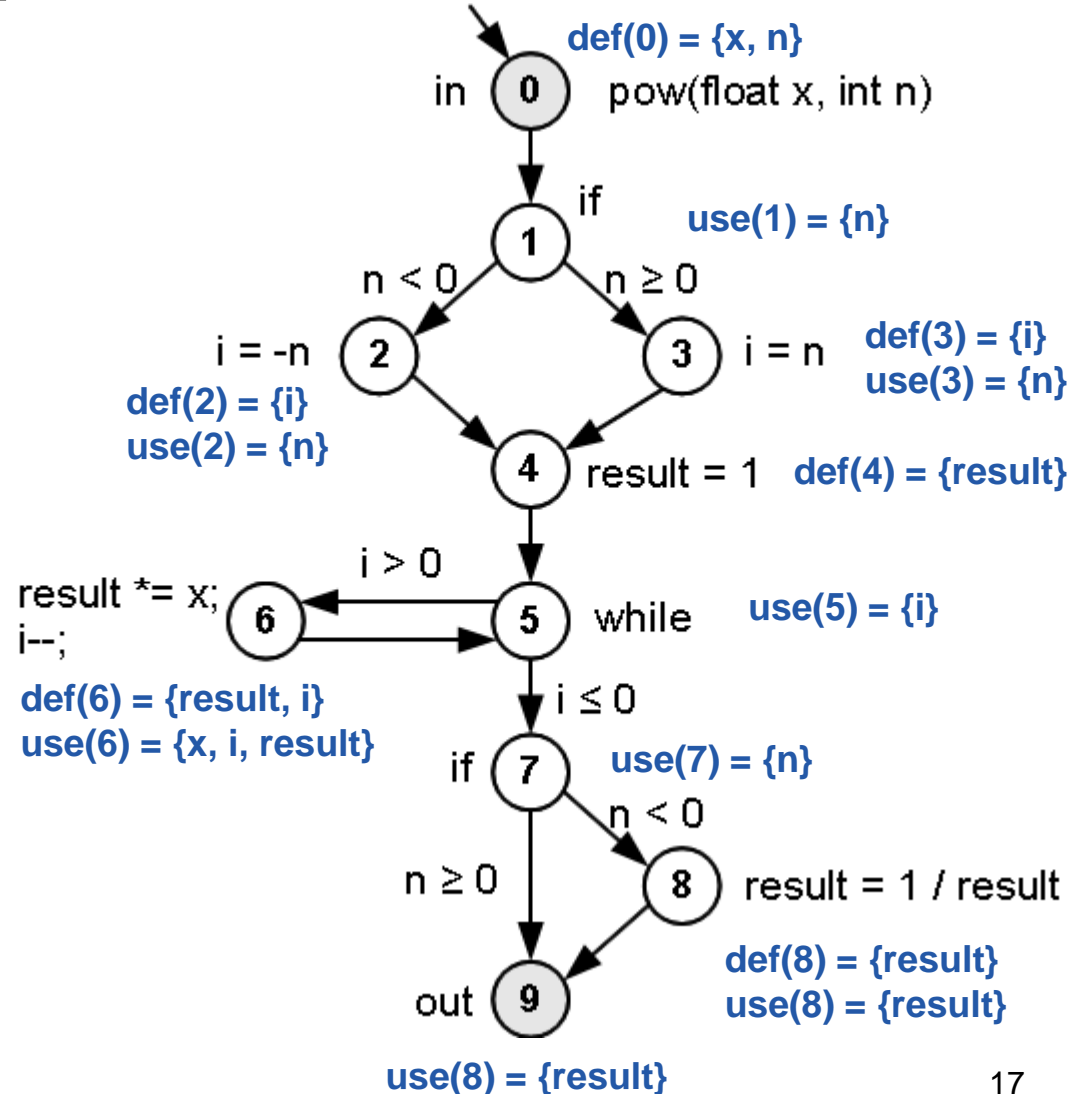


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$   
 $= \{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i)$   
 $= \{(2, 4, 5)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

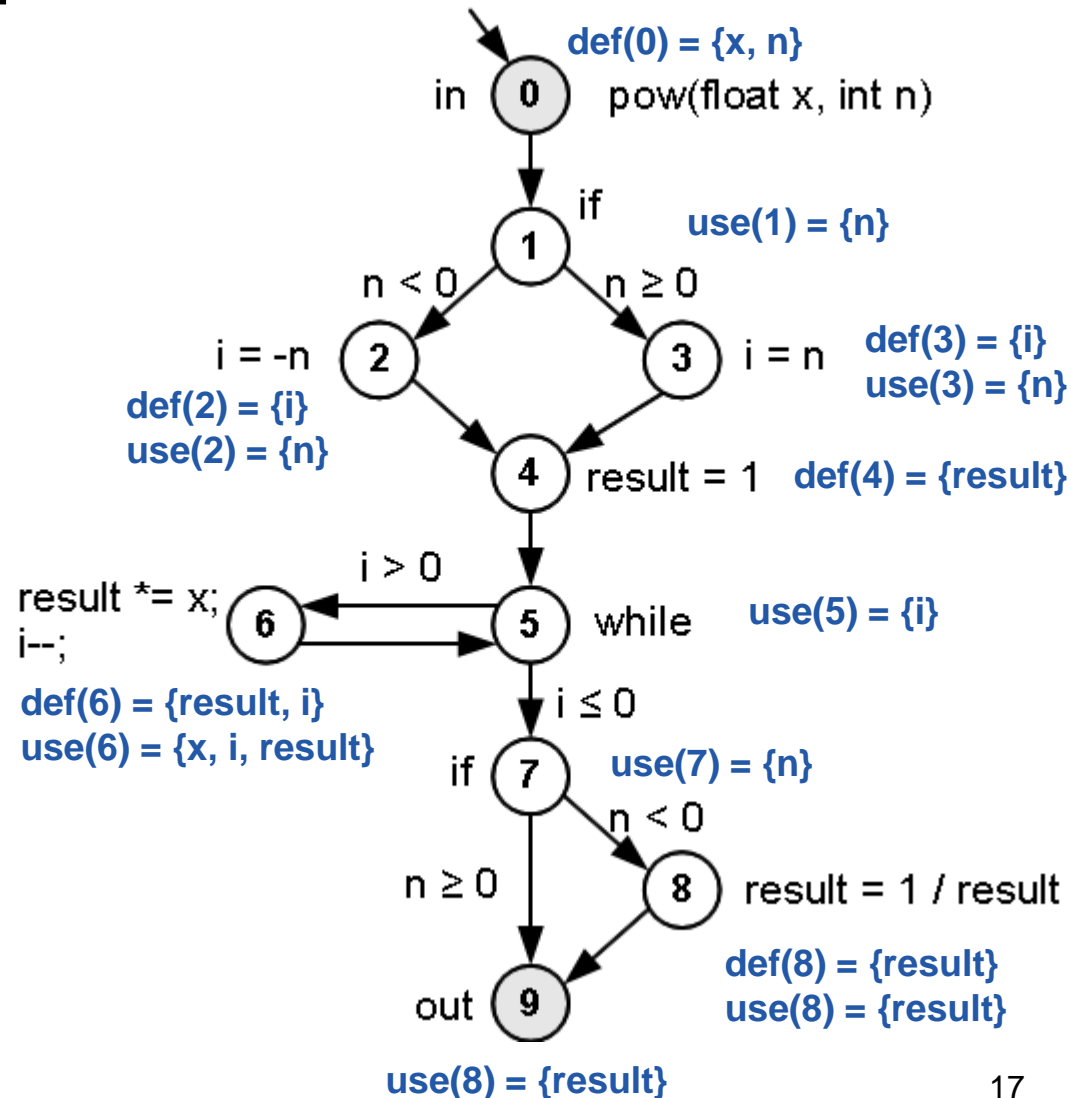
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

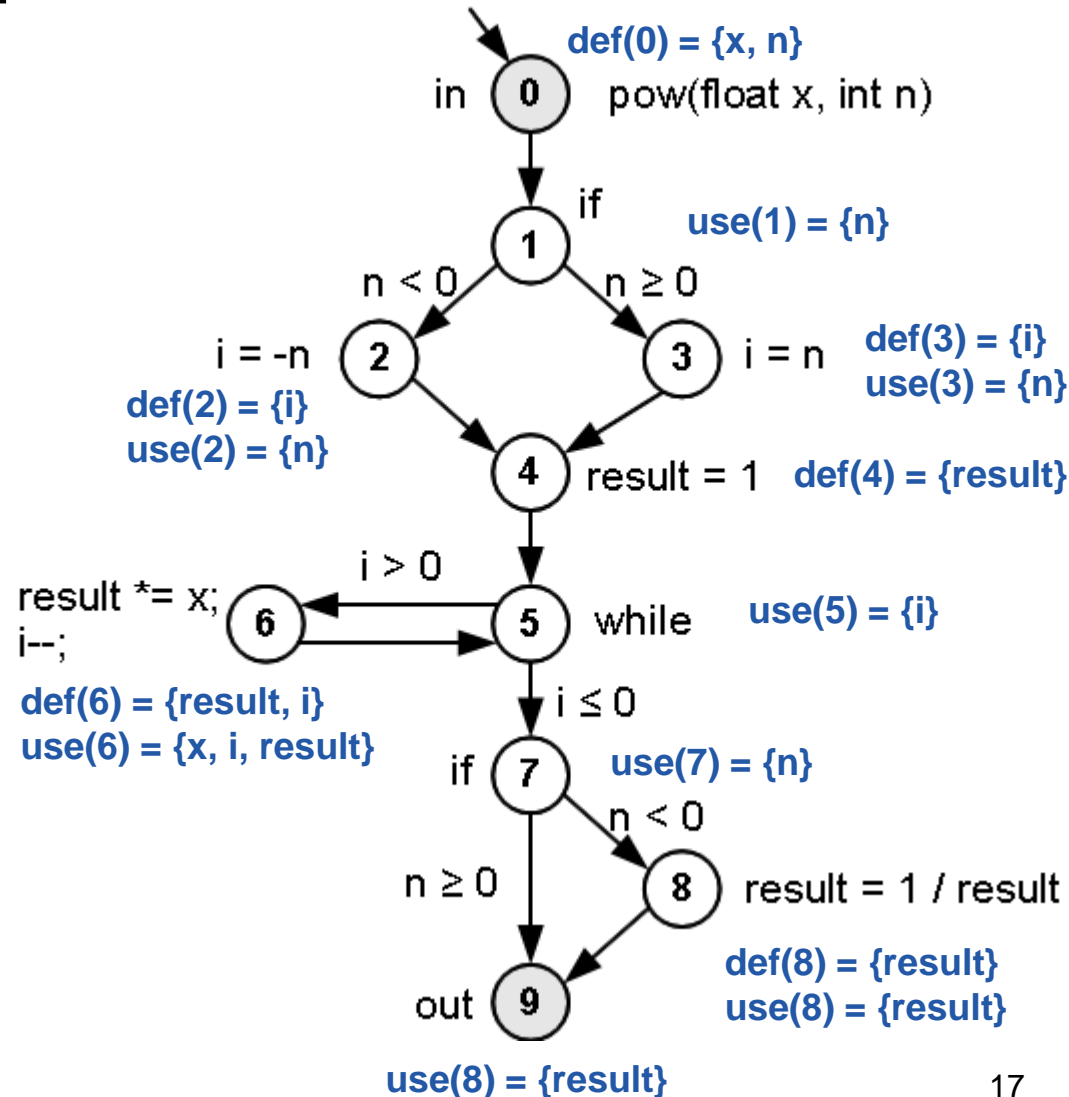
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

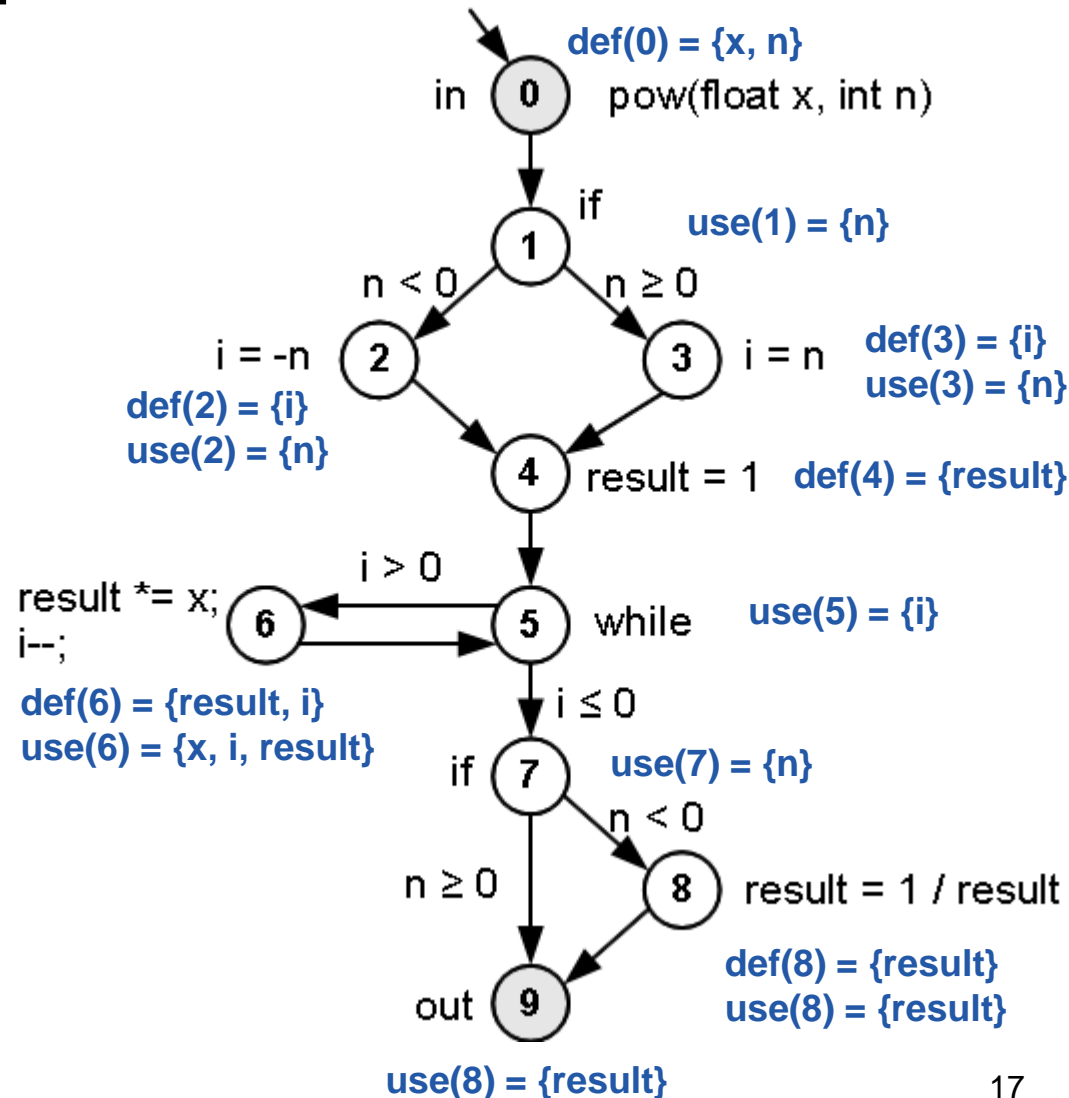
$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

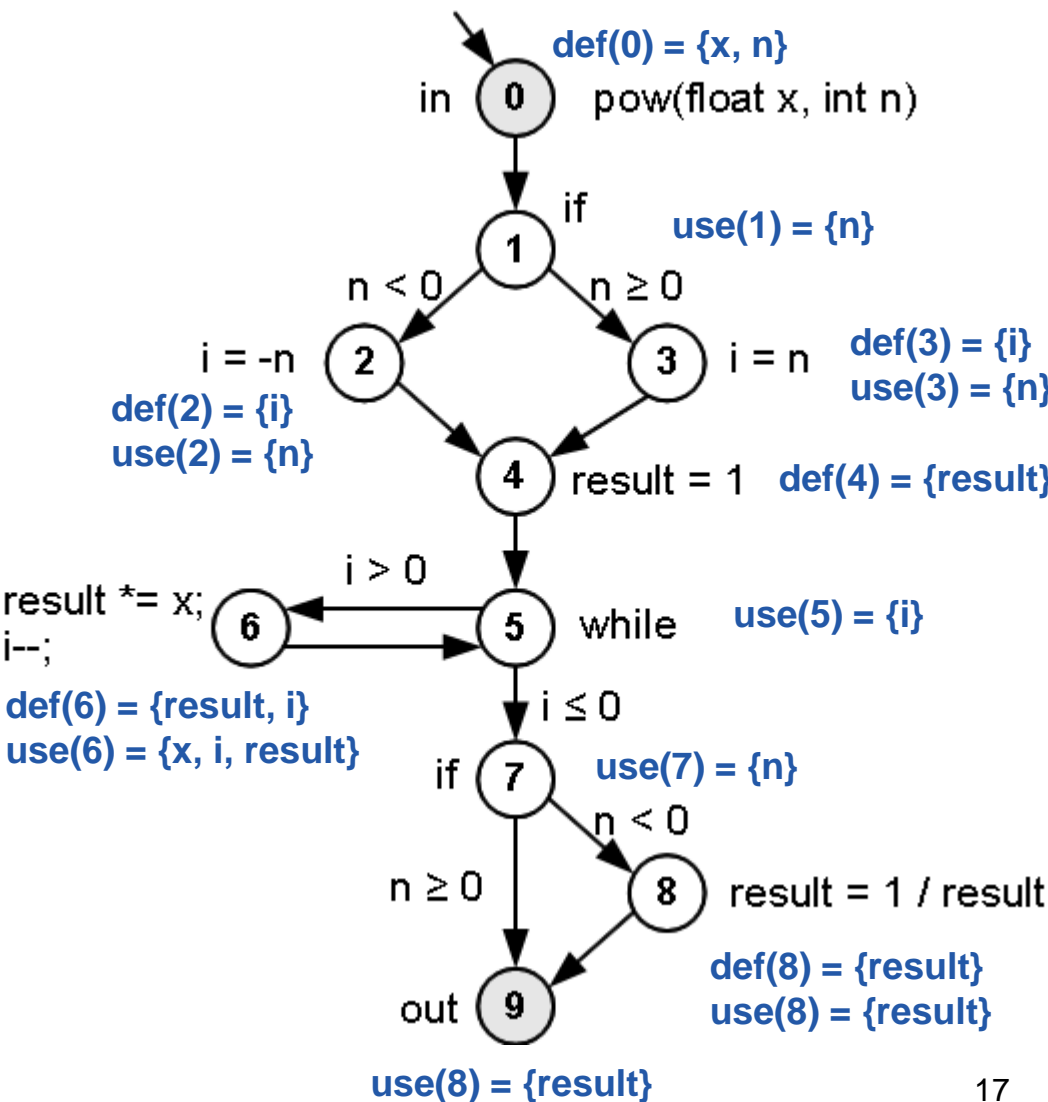
$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), (4, 5, 7, 8), (4, 5, 7, 9)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

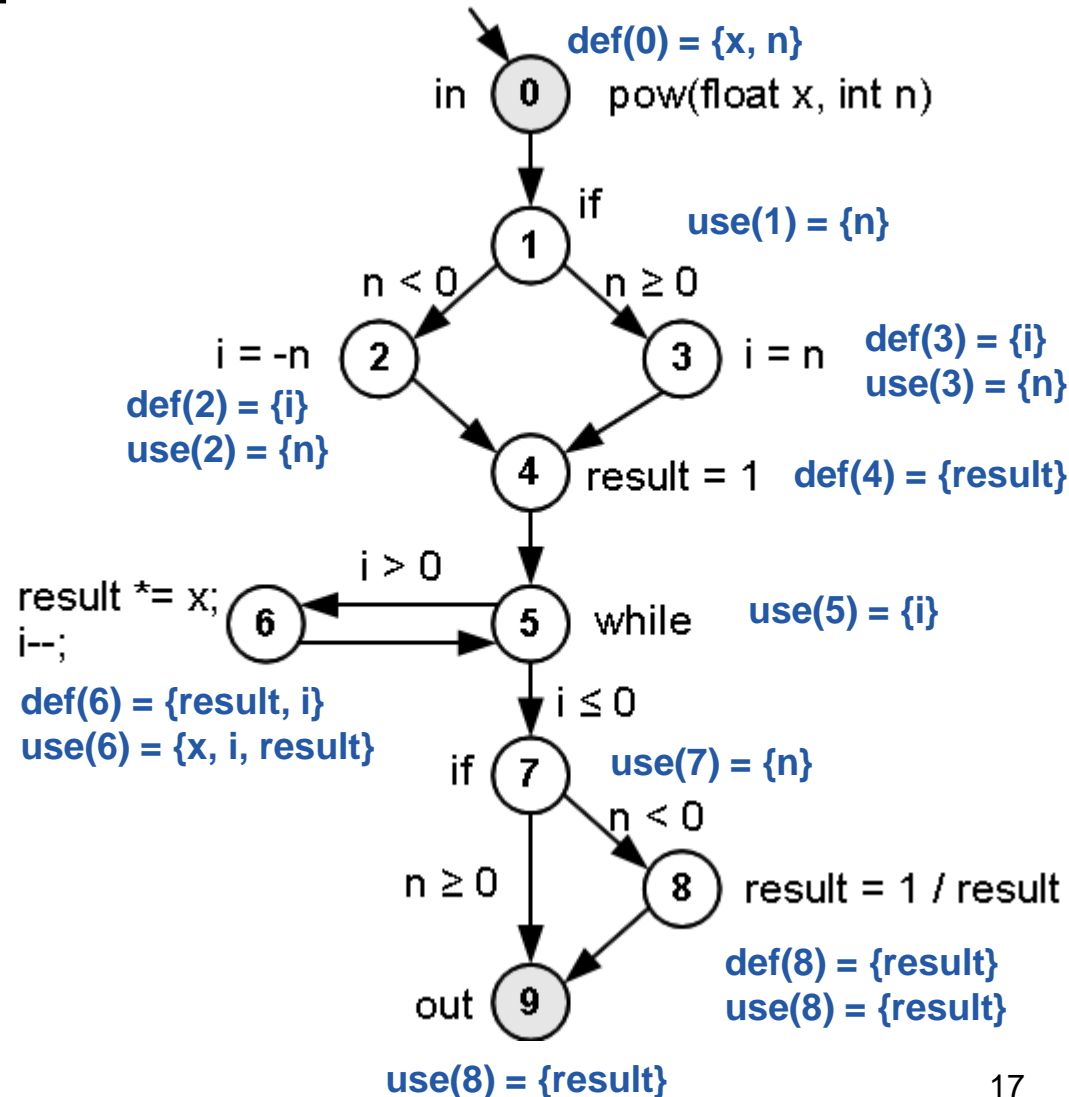
$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

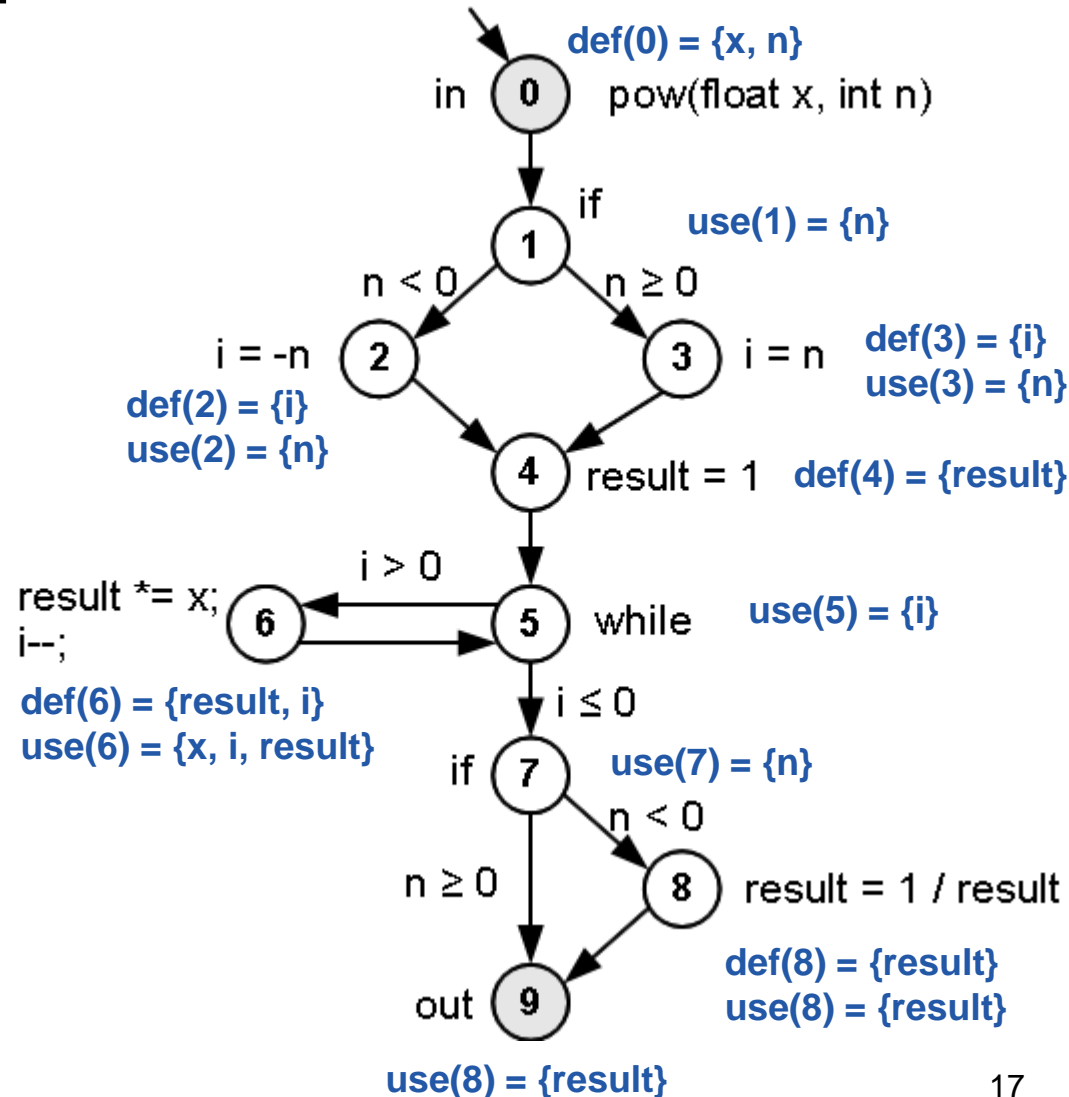
$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$

$du(6, i)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

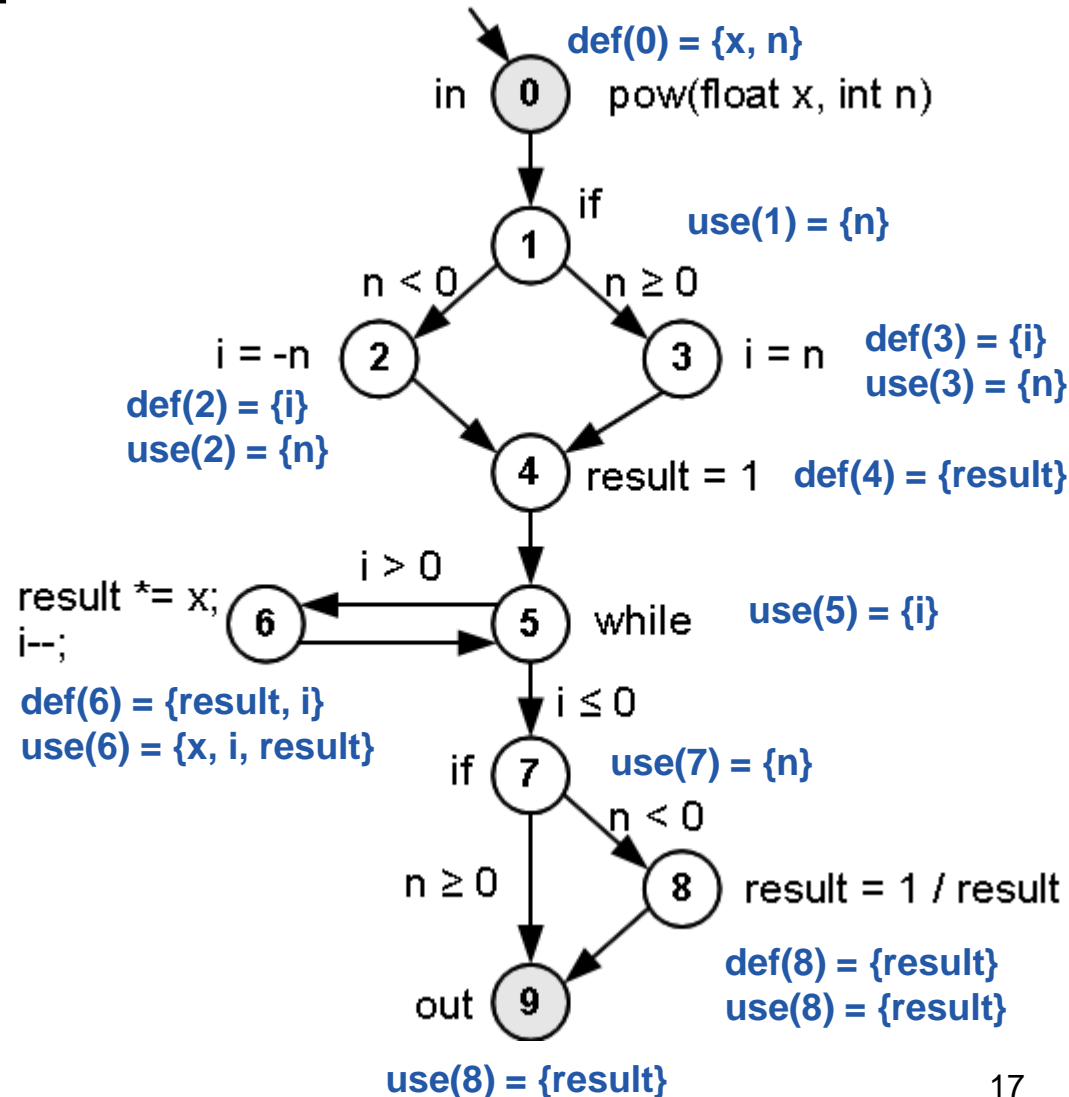
$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$

$du(6, i)$   
=  $\{(6, 5)\}$







# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

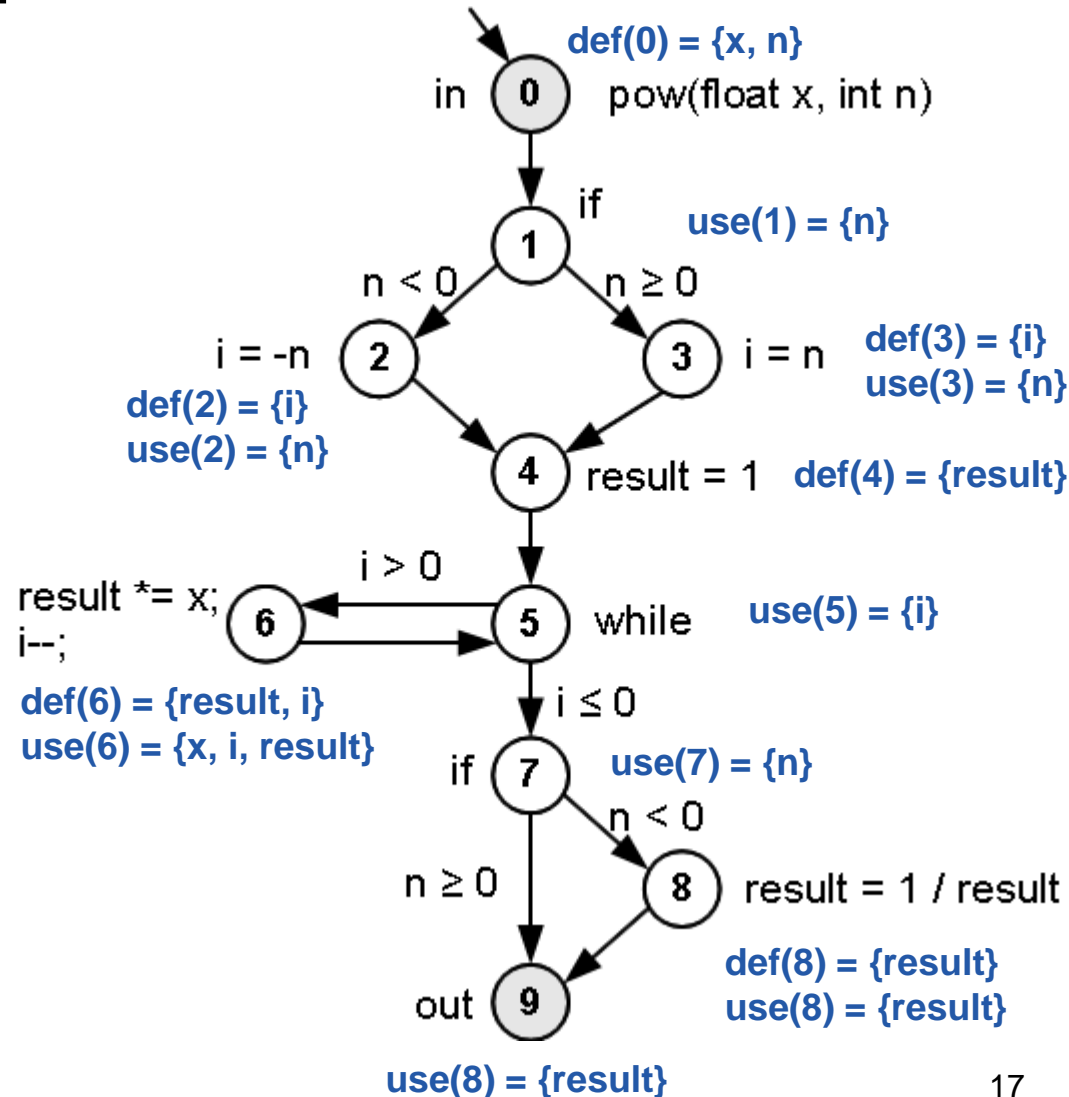
$du(2, i)$   
=  $\{(2, 4, 5)\}$

$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$

$du(6, i)$   
=  $\{(6, 5)\}$

$du(6, result)$

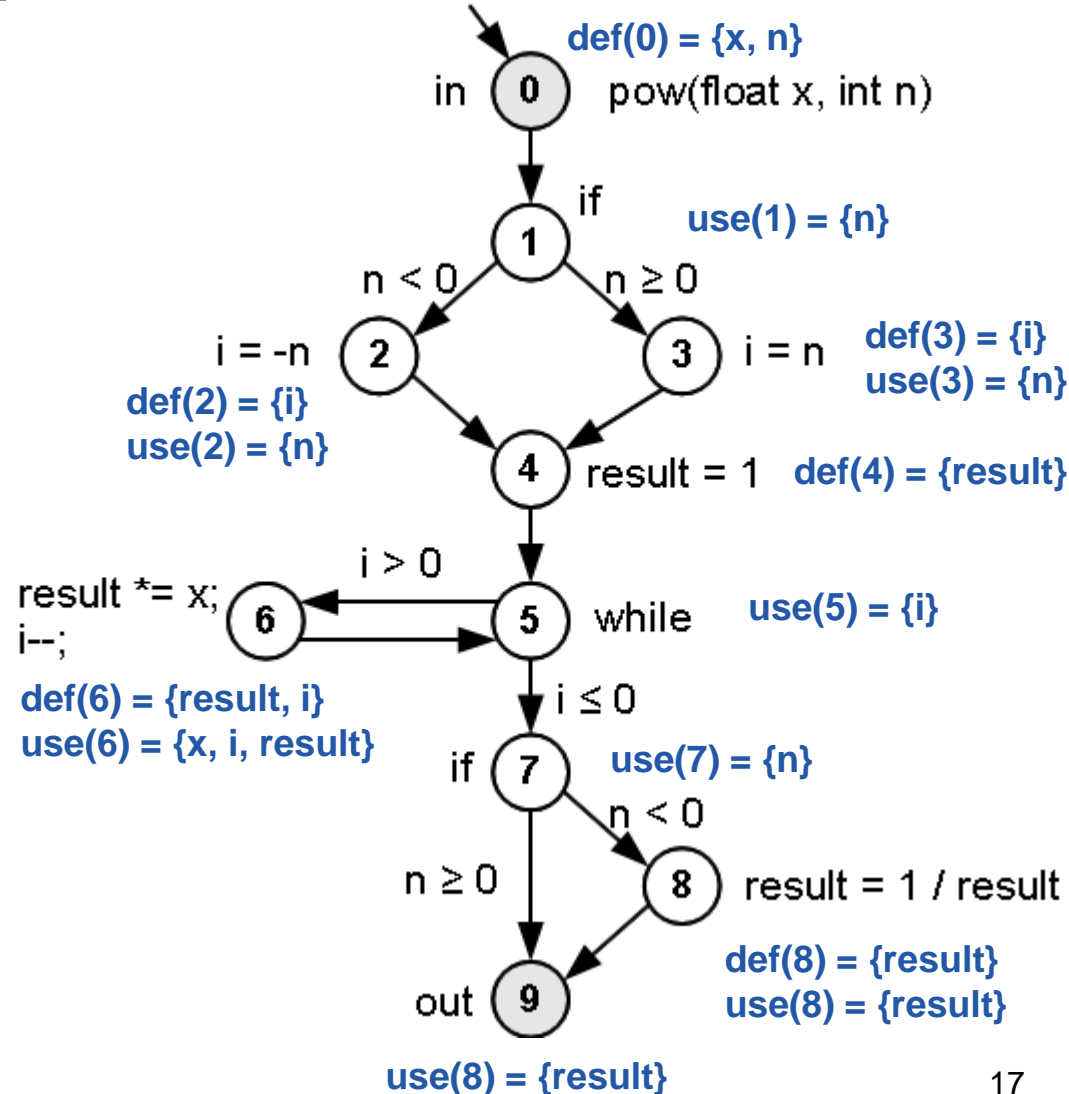


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$   
 $= \{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, i)$   
 $= \{(3, 4, 5)\}$   
 $du(4, result)$   
 $= \{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$   
 $du(6, i)$   
 $= \{(6, 5)\}$   
 $du(6, result)$   
 $= \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

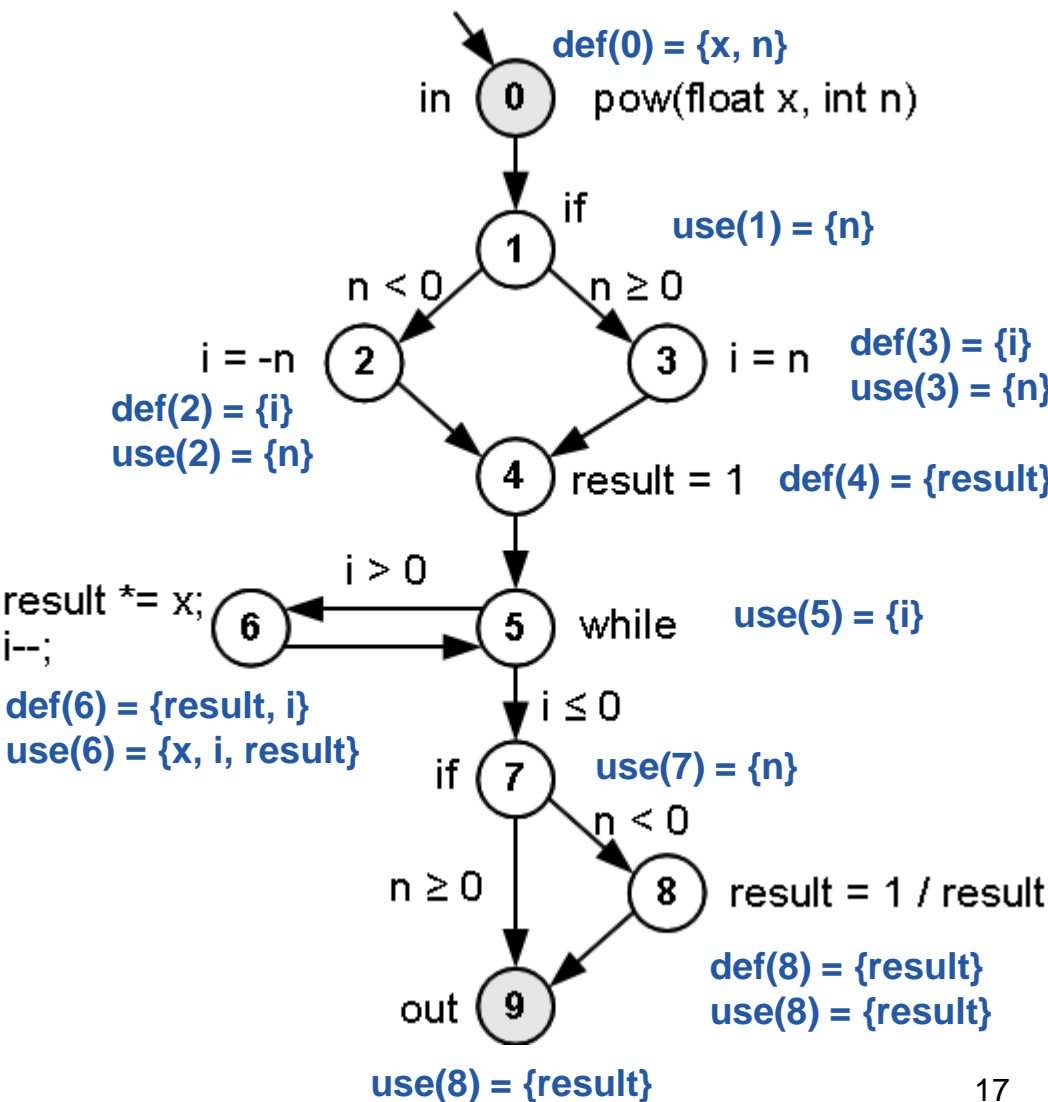
$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$

$du(6, i)$   
=  $\{(6, 5)\}$

$du(6, result)$   
=  $\{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$

$du(8, result)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$

$du(0, n)$   
=  $\{(0, 1), (0, 1, 2, 4, 5, 7), (0, 1, 3, 4, 5, 7)\}$

$du(2, i)$   
=  $\{(2, 4, 5)\}$

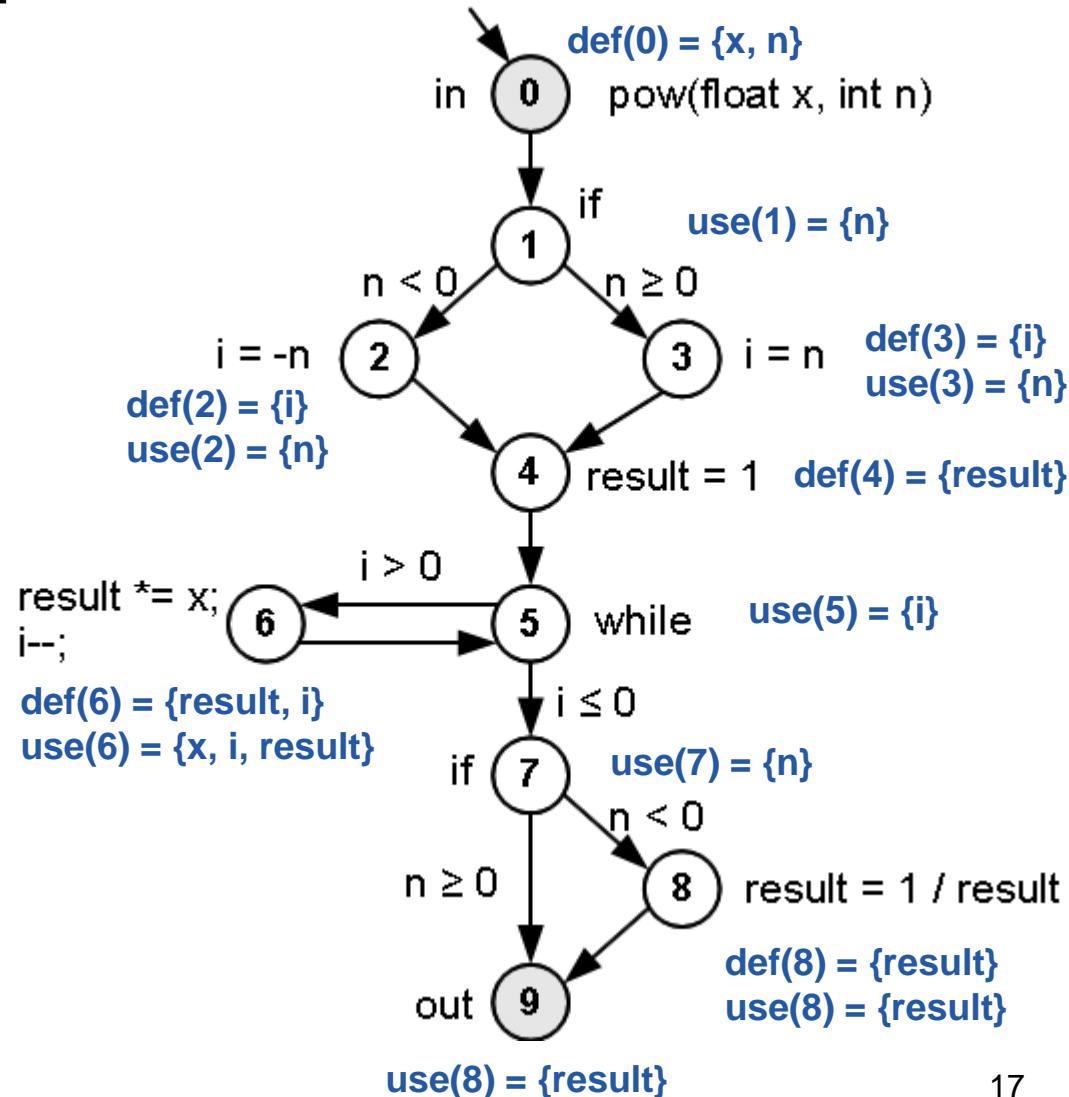
$du(3, i)$   
=  $\{(3, 4, 5)\}$

$du(4, result)$   
=  $\{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$

$du(6, i)$   
=  $\{(6, 5)\}$

$du(6, result)$   
=  $\{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$

$du(8, result)$   
=  $\{(8, 9)\}$

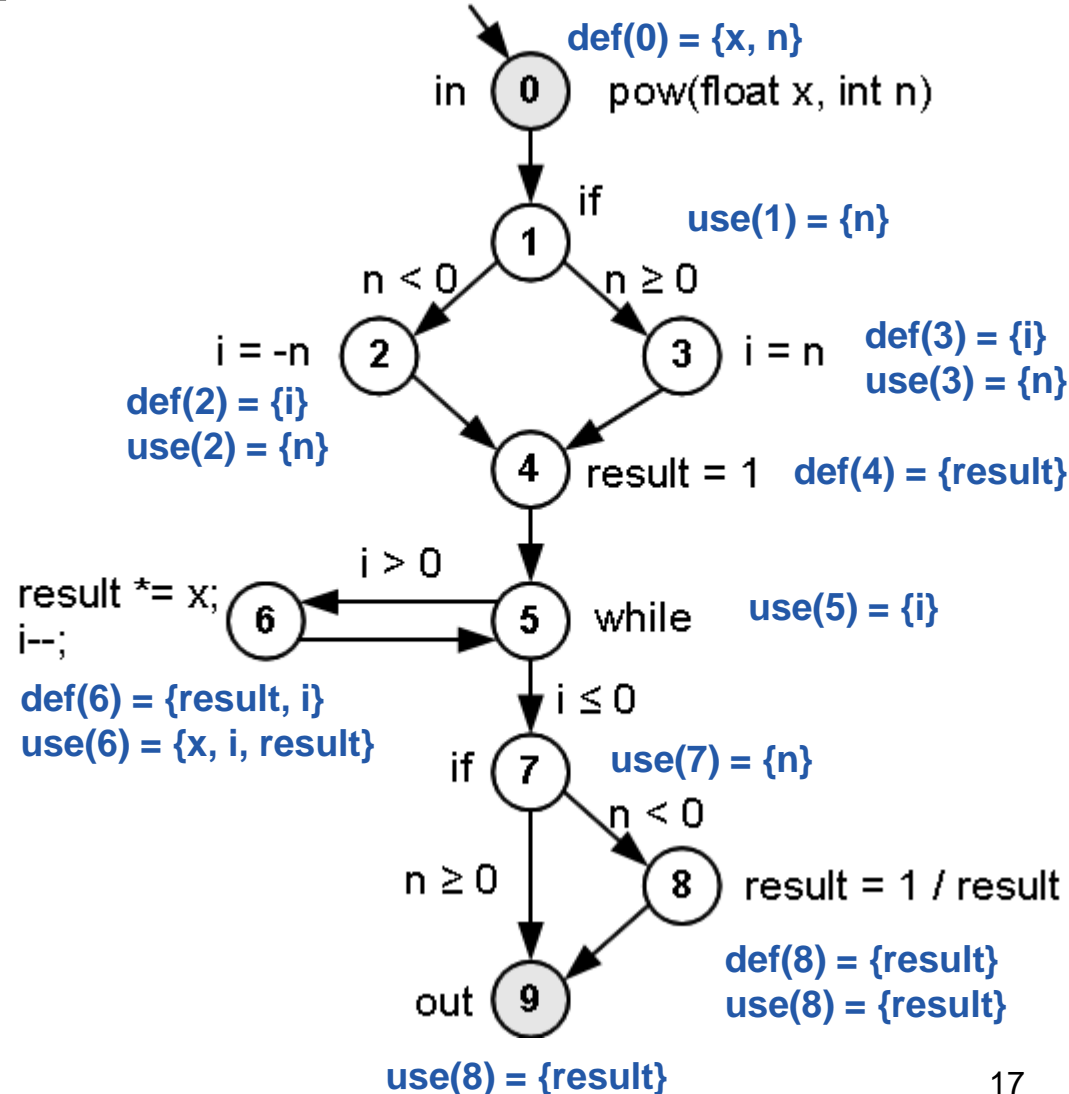


# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n)$   
 $= \{(0, 1), \text{~~(0, 1, 2, 4, 5, 7)~~, (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, i)$   
 $= \{(3, 4, 5)\}$   
 $du(4, \text{result})$   
 $= \{(4, 5, 6), \text{~~(4, 5, 7, 8)~~, (4, 5, 7, 9)\}$   
 $du(6, i)$   
 $= \{(6, 5)\}$   
 $du(6, \text{result})$   
 $= \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$   
 $du(8, \text{result})$   
 $= \{(8, 9)\}$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

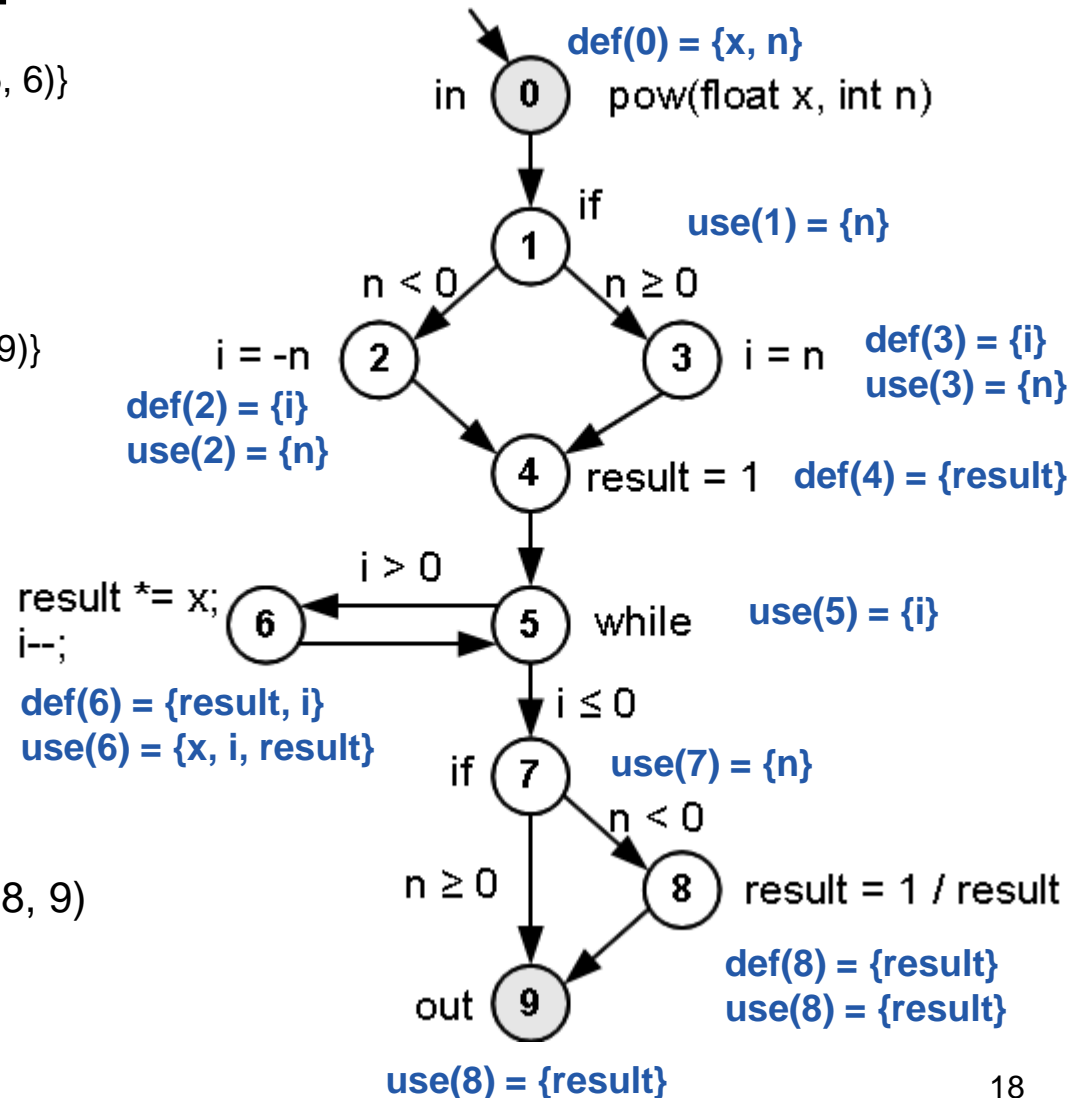
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$	$= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$
$du(0, n)$	$= \{(0, 1), (0, 1, 3, 4, 5, 7)\}$
$du(2, i)$	$= \{(2, 4, 5)\}$
$du(3, i)$	$= \{(3, 4, 5)\}$
$du(4, result)$	$= \{(4, 5, 6), (4, 5, 7, 9)\}$
$du(6, i)$	$= \{(6, 5)\}$
$du(6, result)$	$= \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$
$du(8, result)$	$= \{(8, 9)\}$

- **All-Definitions:**

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

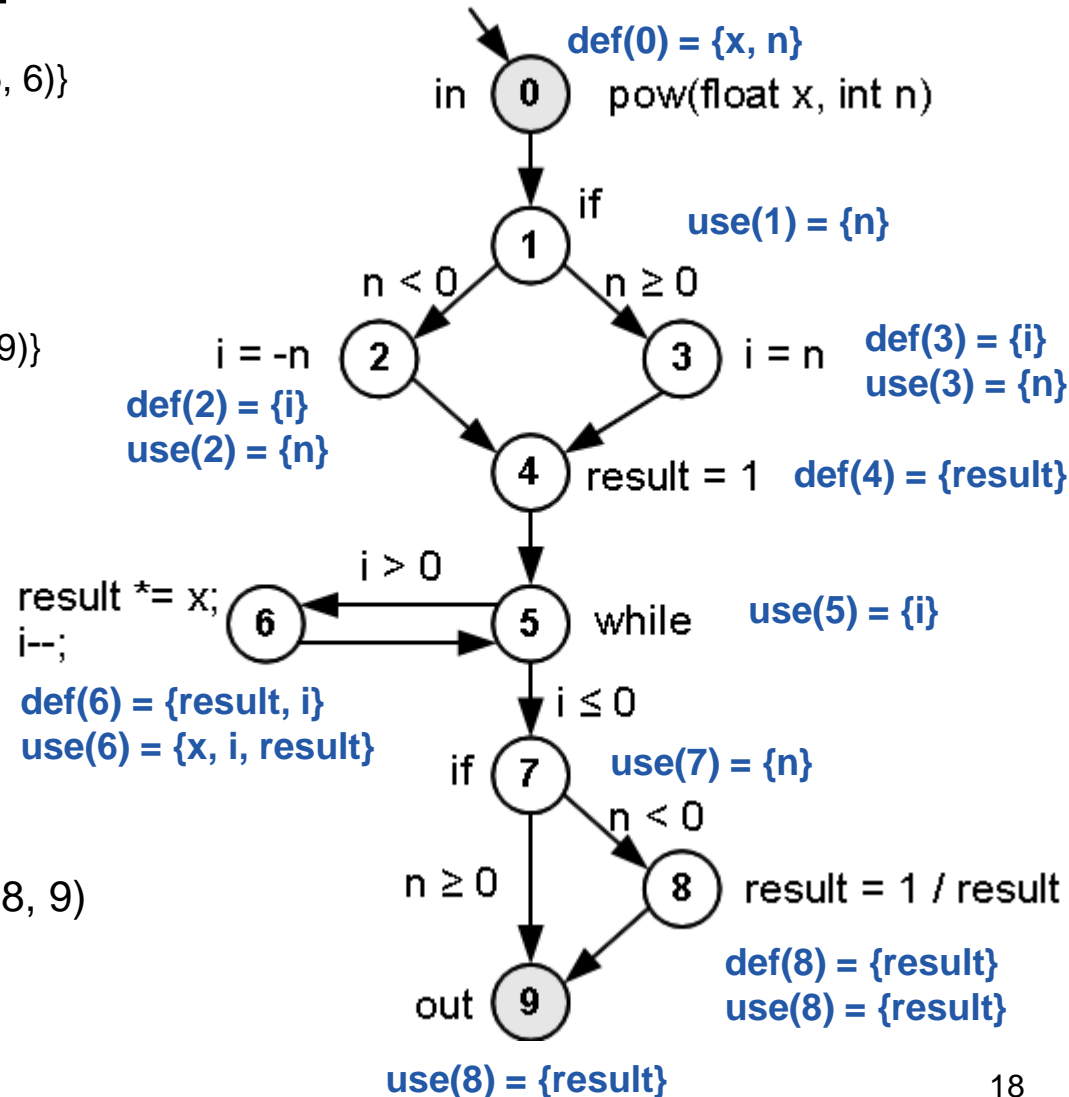
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$	$= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$
$du(0, n)$	$= \{(0, 1), (0, 1, 3, 4, 5, 7)\}$
$du(2, i)$	$= \{(2, 4, 5)\}$
$du(3, i)$	$= \{(3, 4, 5)\}$
$du(4, result)$	$= \{(4, 5, 6), (4, 5, 7, 9)\}$
$du(6, i)$	$= \{(6, 5)\}$
$du(6, result)$	$= \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$
$du(8, result)$	$= \{(8, 9)\}$

- All-Definitions:

- Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

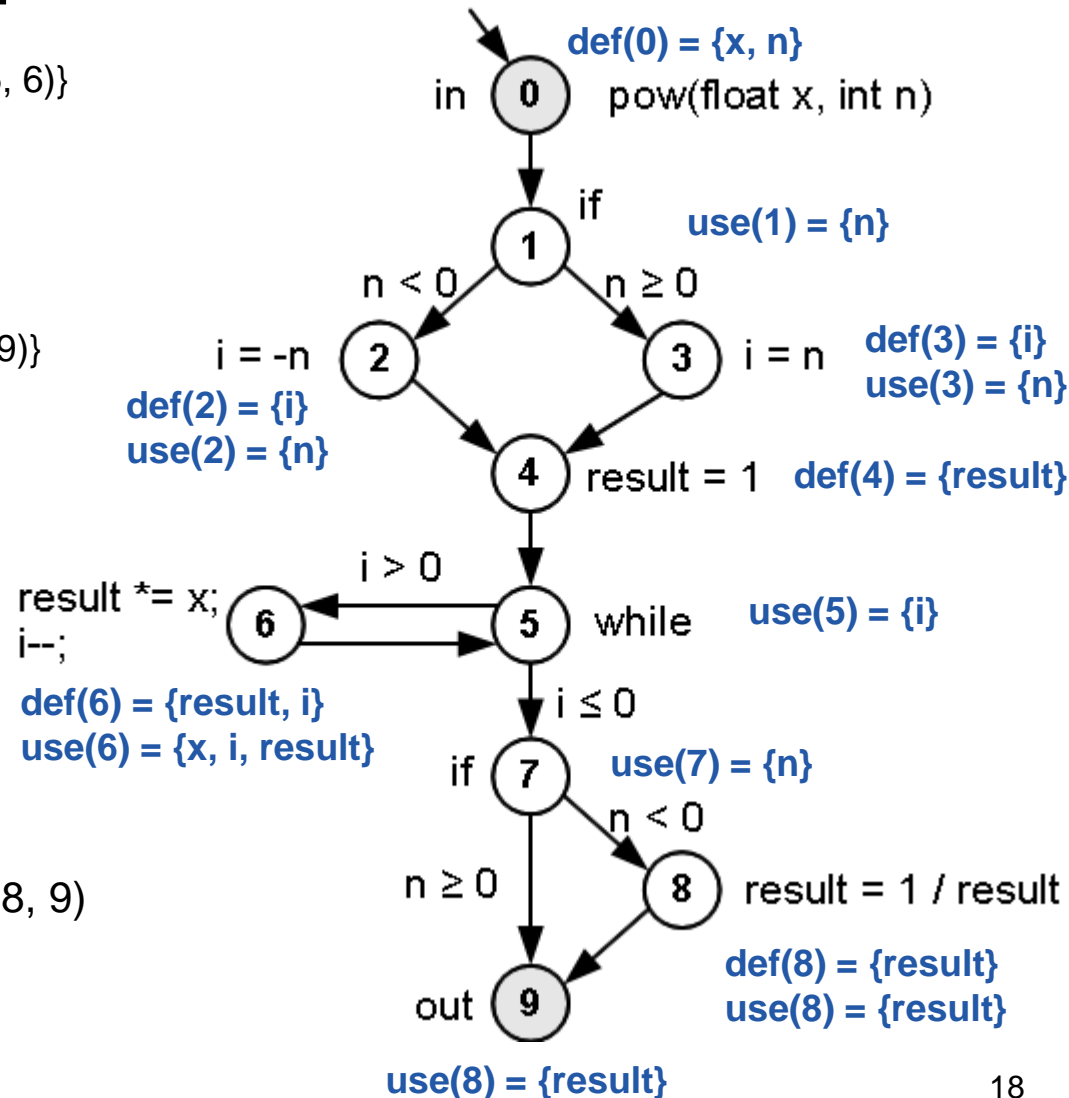
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n) = \{(0, 1), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(2, 4, 5)\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(4, 5, 6), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(6, 5)\}$   
 $du(6, result) = \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(8, 9)\}$

- All-Definitions:

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$







# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

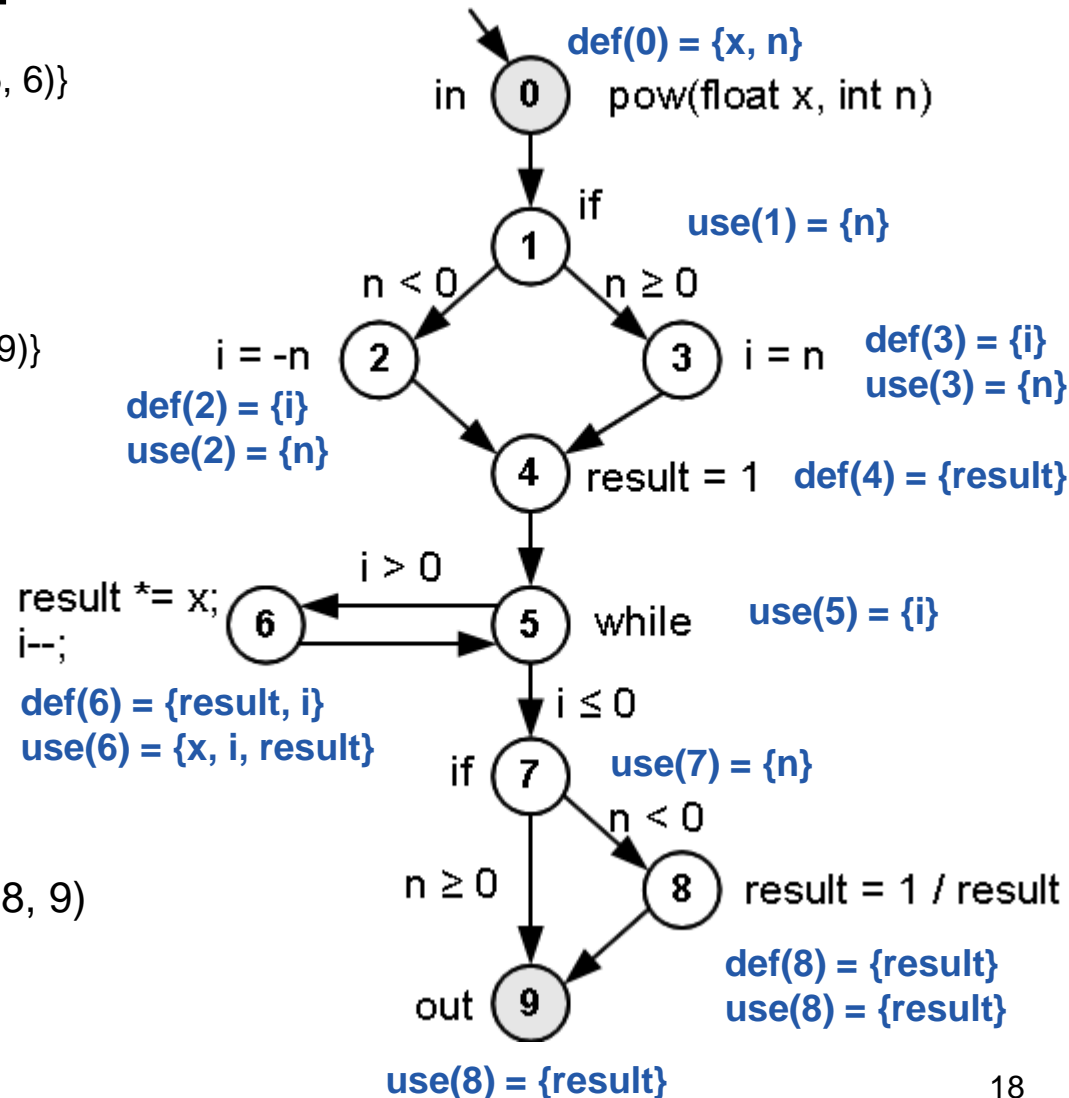
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n) = \{(0, 1), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(2, 4, 5)\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(4, 5, 6), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(6, 5)\}$   
 $du(6, result) = \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(8, 9)\}$

- All-Definitions:

- Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

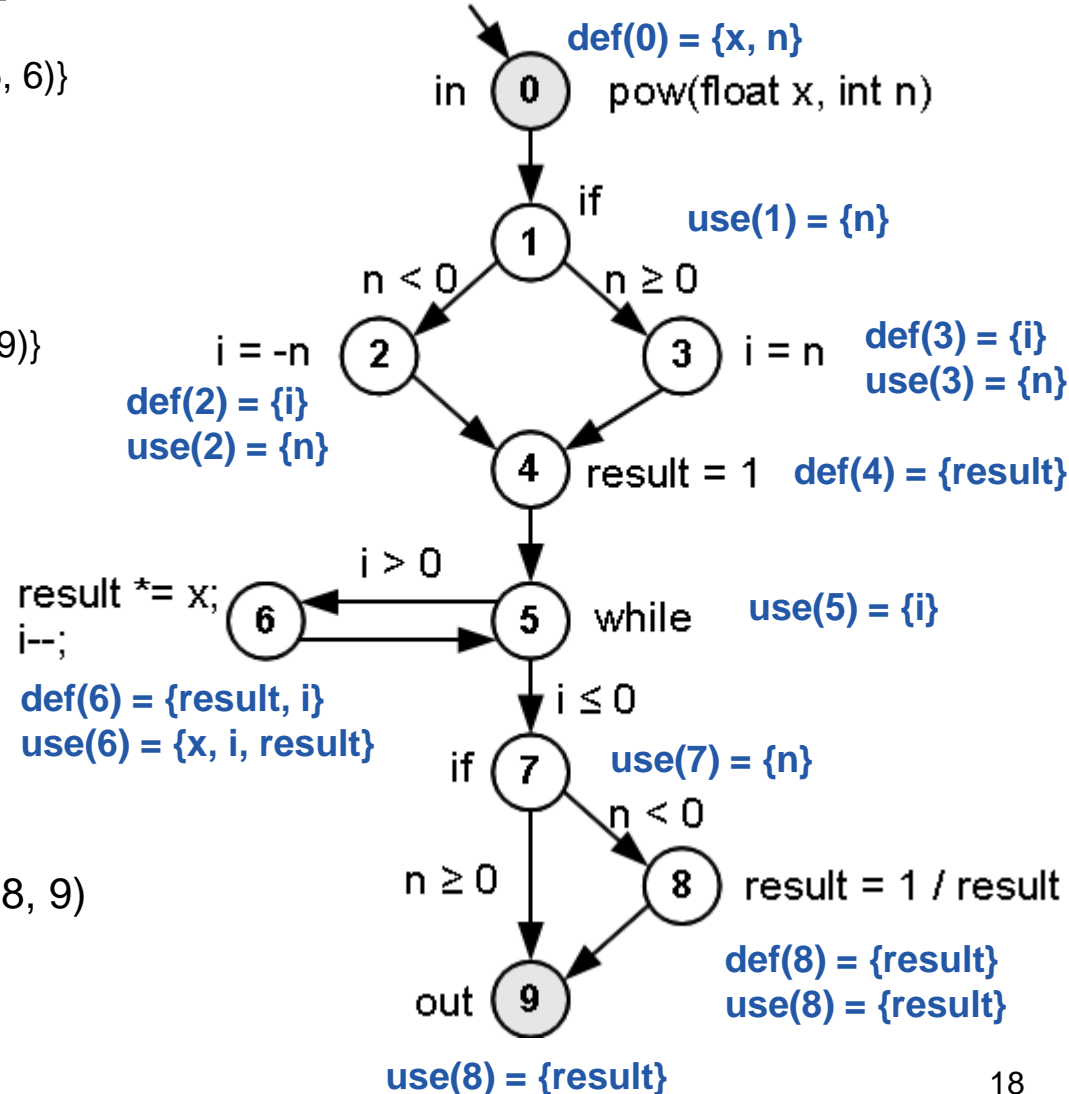
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\overline{(0, 1, 2, 4, 5, 6)}, (0, 1, 3, 4, 5, 6))\}$   
 $du(0, n) = \{(\overline{(0, 1)}, (0, 1, 3, 4, 5, 7))\}$   
 $du(2, i) = \{(\overline{(2, 4, 5)})\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(\overline{(4, 5, 6)}, (4, 5, 7, 9))\}$   
 $du(6, i) = \{(6, 5)\}$   
 $du(6, result) = \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(8, 9)\}$

- All-Definitions:

- Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

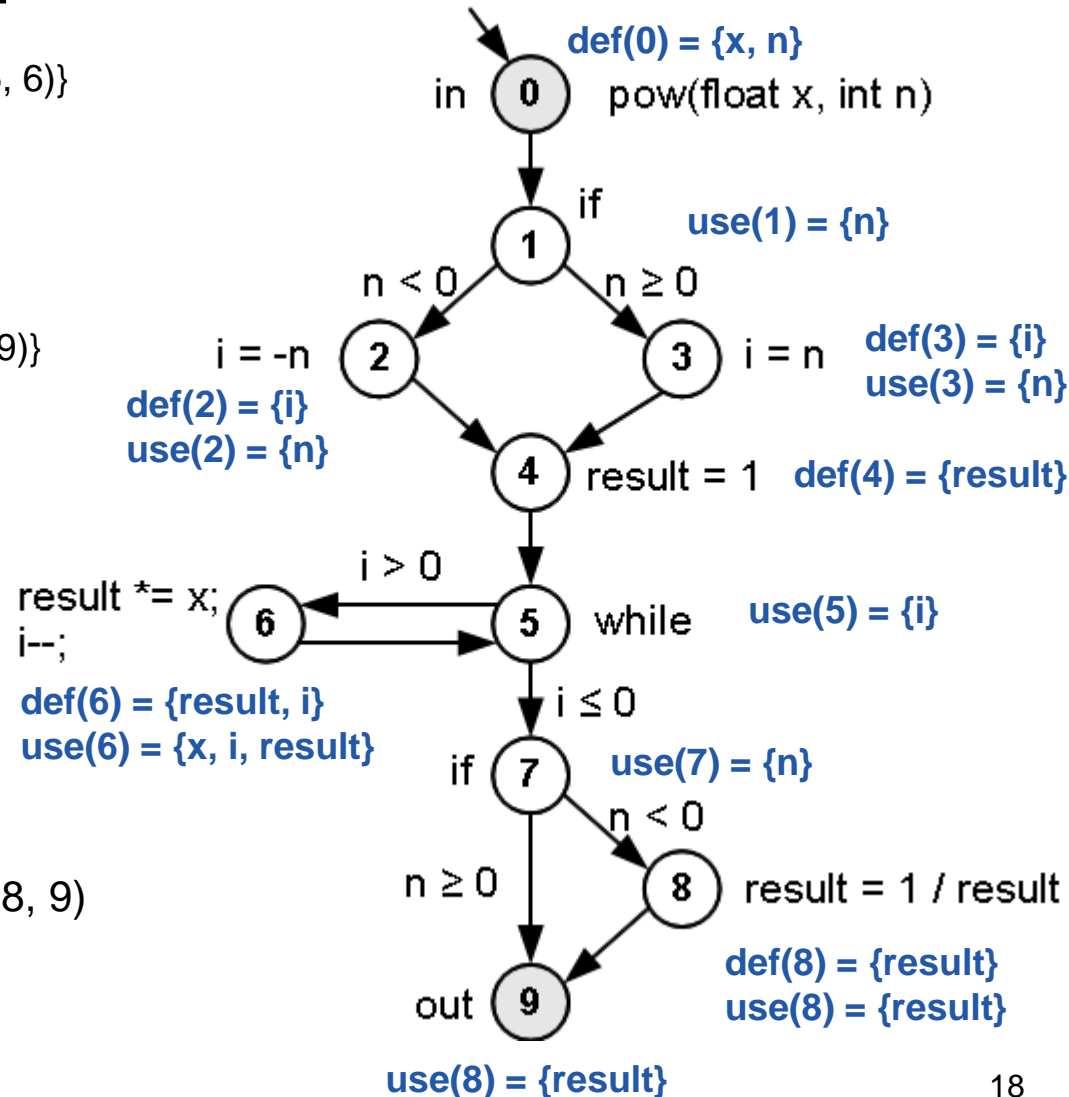
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x)$	$= \{(\overline{0, 1, 2, 4, 5, 6}), (0, 1, 3, 4, 5, 6)\}$
$du(0, n)$	$= \{(\overline{0, 1}), (0, 1, 3, 4, 5, 7)\}$
$du(2, i)$	$= \{(\overline{2, 4, 5})\}$
$du(3, i)$	$= \{(3, 4, 5)\}$
$du(4, result)$	$= \{(\overline{4, 5, 6}), (4, 5, 7, 9)\}$
$du(6, i)$	$= \{(\overline{6, 5})\}$
$du(6, result)$	$= \{(6, 5, 6), (6, 5, 7, 8), (6, 5, 7, 9)\}$
$du(8, result)$	$= \{(8, 9)\}$

- All-Definitions:**

- Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

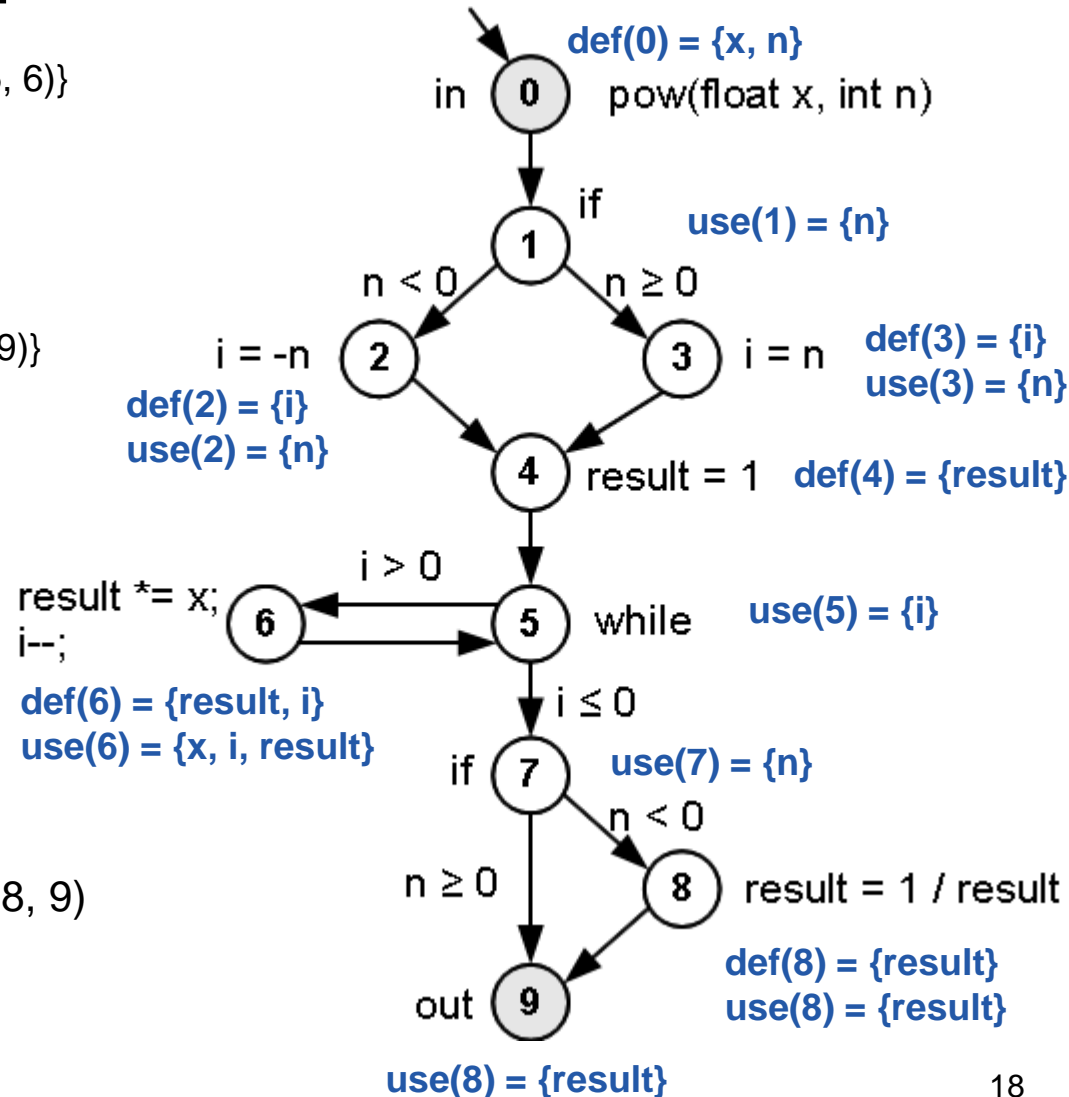
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\overline{0, 1, 2, 4, 5, 6}), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n) = \{(\overline{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\overline{2, 4, 5})\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(\overline{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\overline{6, 5})\}$   
 $du(6, result) = \{(\overline{6, 5, 6}), (\overline{6, 5, 7, 8}), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(8, 9)\}$

- All-Definitions:**

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

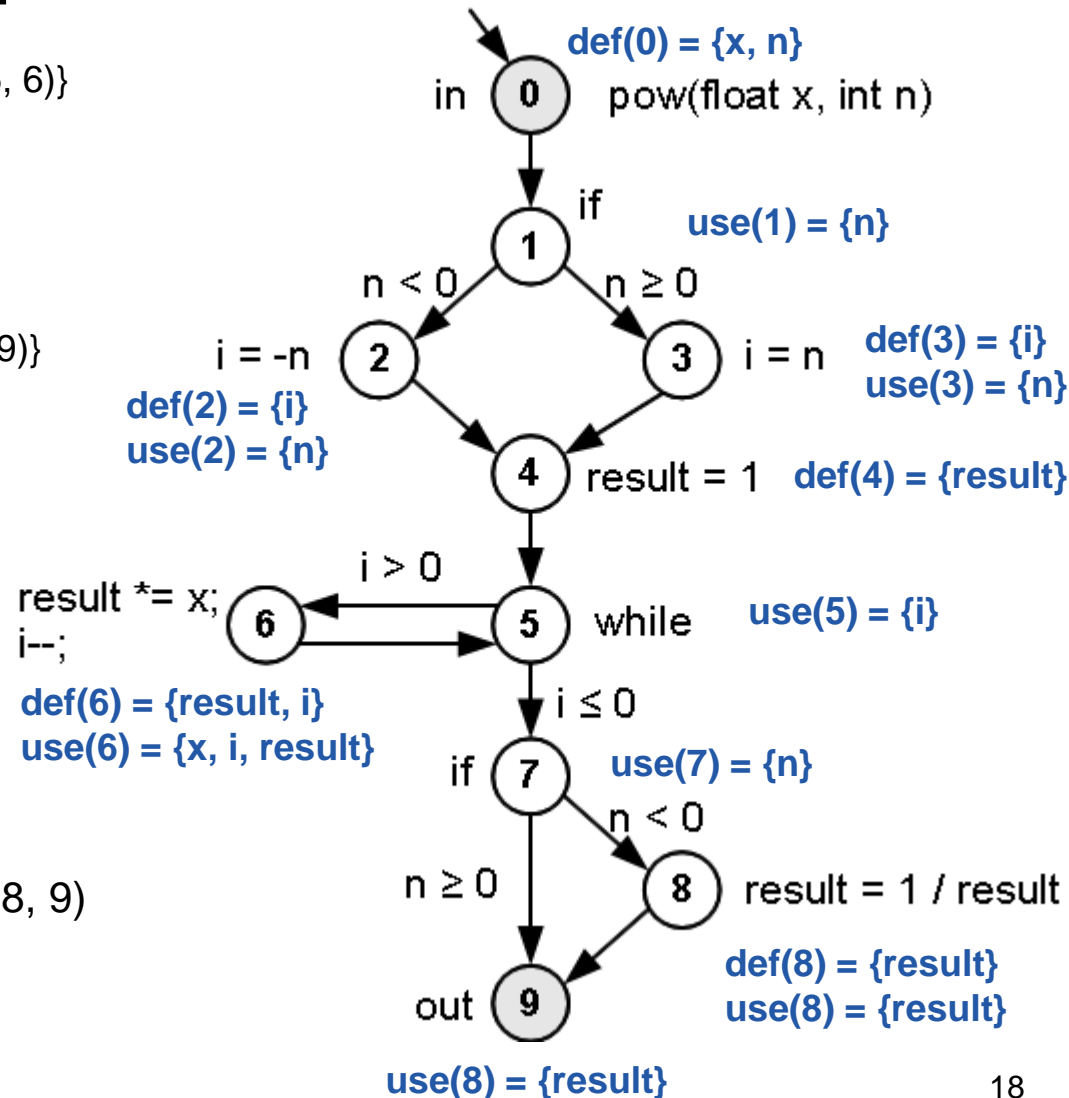
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\overline{0, 1, 2, 4, 5, 6}), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n) = \{(\overline{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\overline{2, 4, 5})\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(\overline{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\overline{6, 5})\}$   
 $du(6, result) = \{(\overline{6, 5, 6}), (\overline{6, 5, 7, 8}), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(\overline{8, 9})\}$

- All-Definitions:

- Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

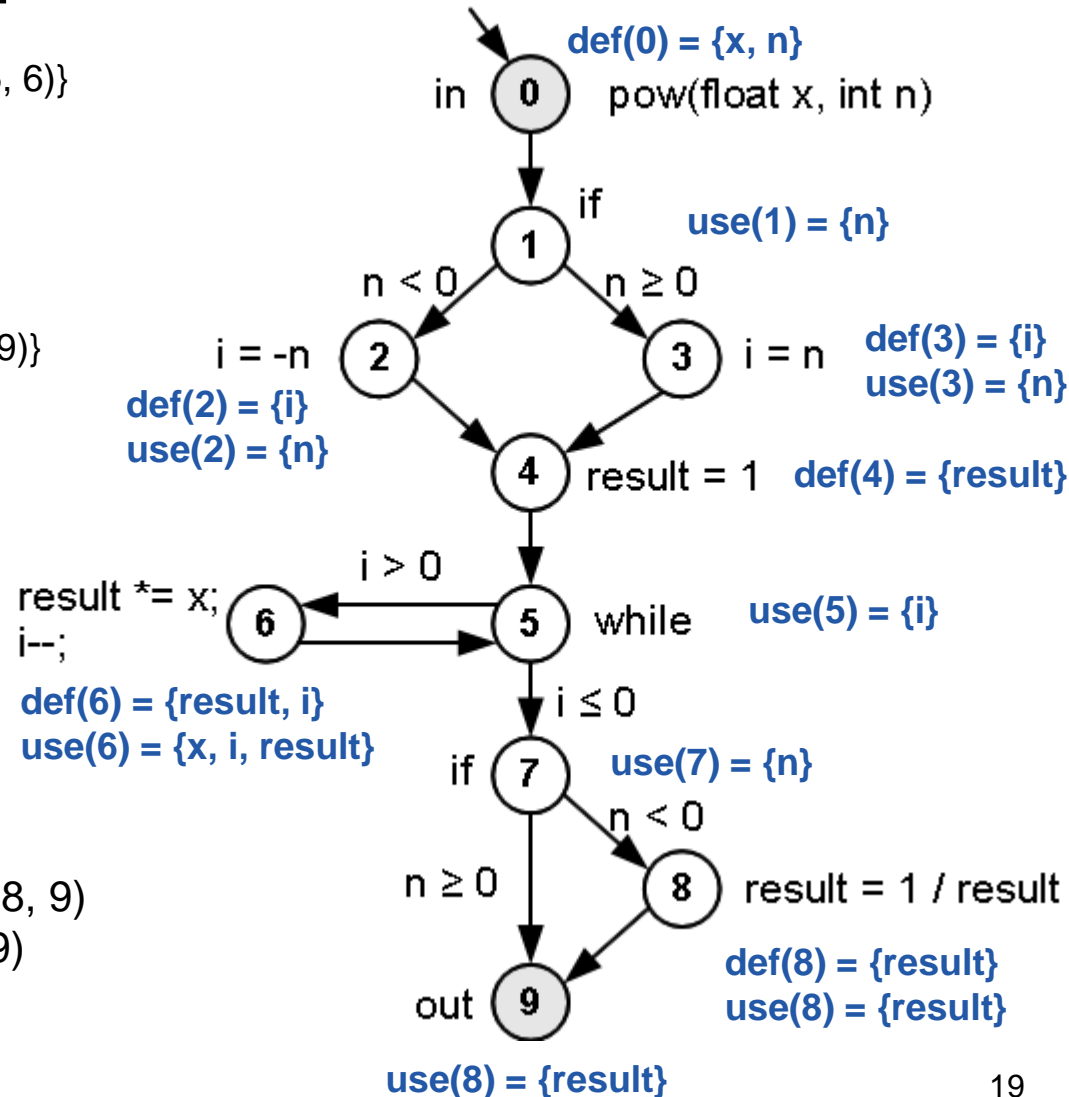
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\overline{0, 1, 2, 4, 5, 6}), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, n) = \{(\overline{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\overline{2, 4, 5})\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(\overline{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\overline{6, 5})\}$   
 $du(6, result) = \{(\overline{6, 5, 6}), (\overline{6, 5, 7, 8}), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(\overline{8, 9})\}$

- All-Definitions:**

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

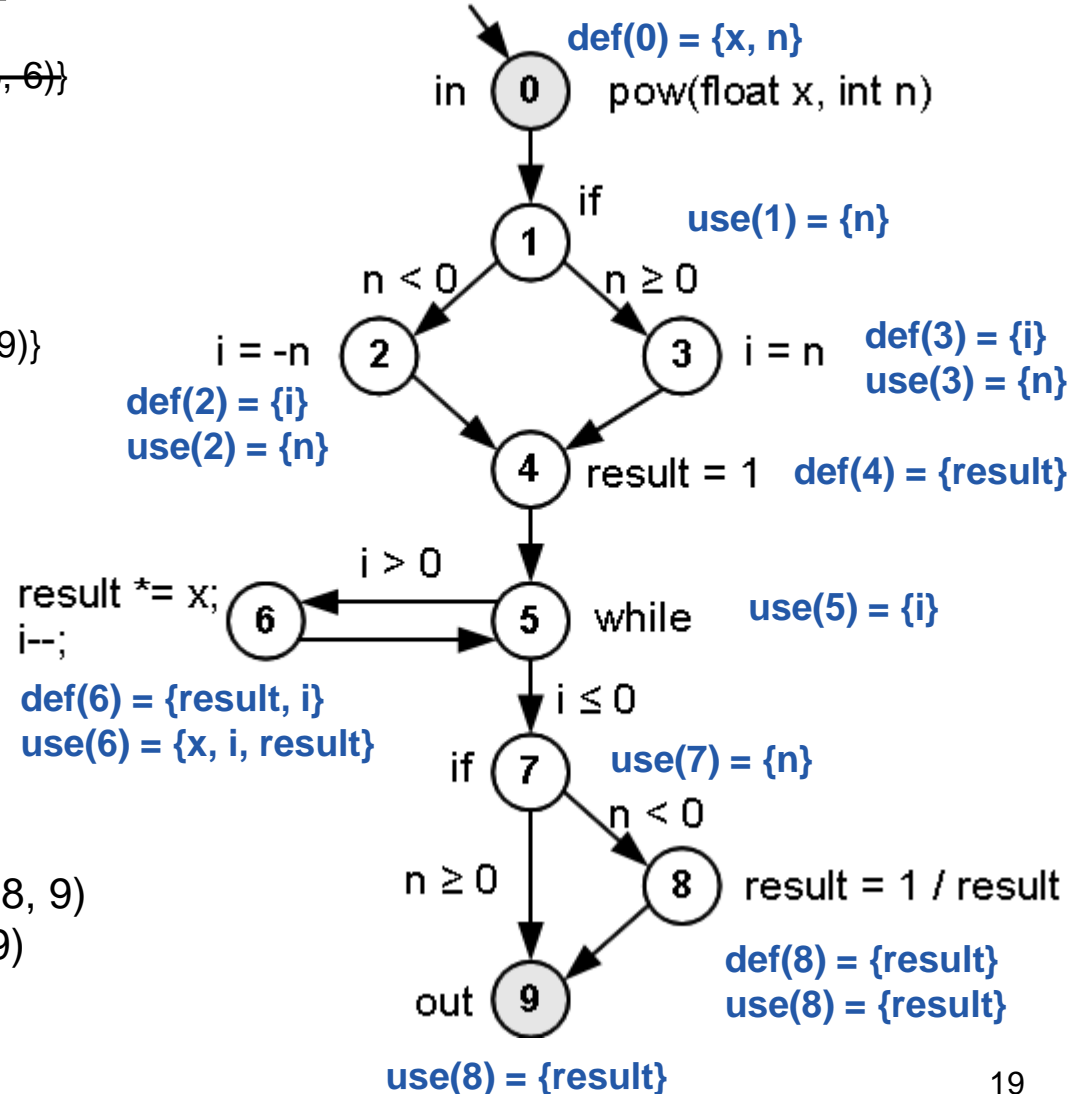
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\cancel{0, 1, 2, 4, 5, 6}), (\cancel{0, 1, 3, 4, 5, 6})\}$   
 $du(0, n) = \{(\cancel{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\cancel{2, 4, 5})\}$   
 $du(3, i) = \{(3, 4, 5)\}$   
 $du(4, result) = \{(\cancel{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\cancel{6, 5})\}$   
 $du(6, result) = \{(\cancel{6, 5, 6}), (\cancel{6, 5, 7, 8}), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(\cancel{8, 9})\}$

- All-Definitions:

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

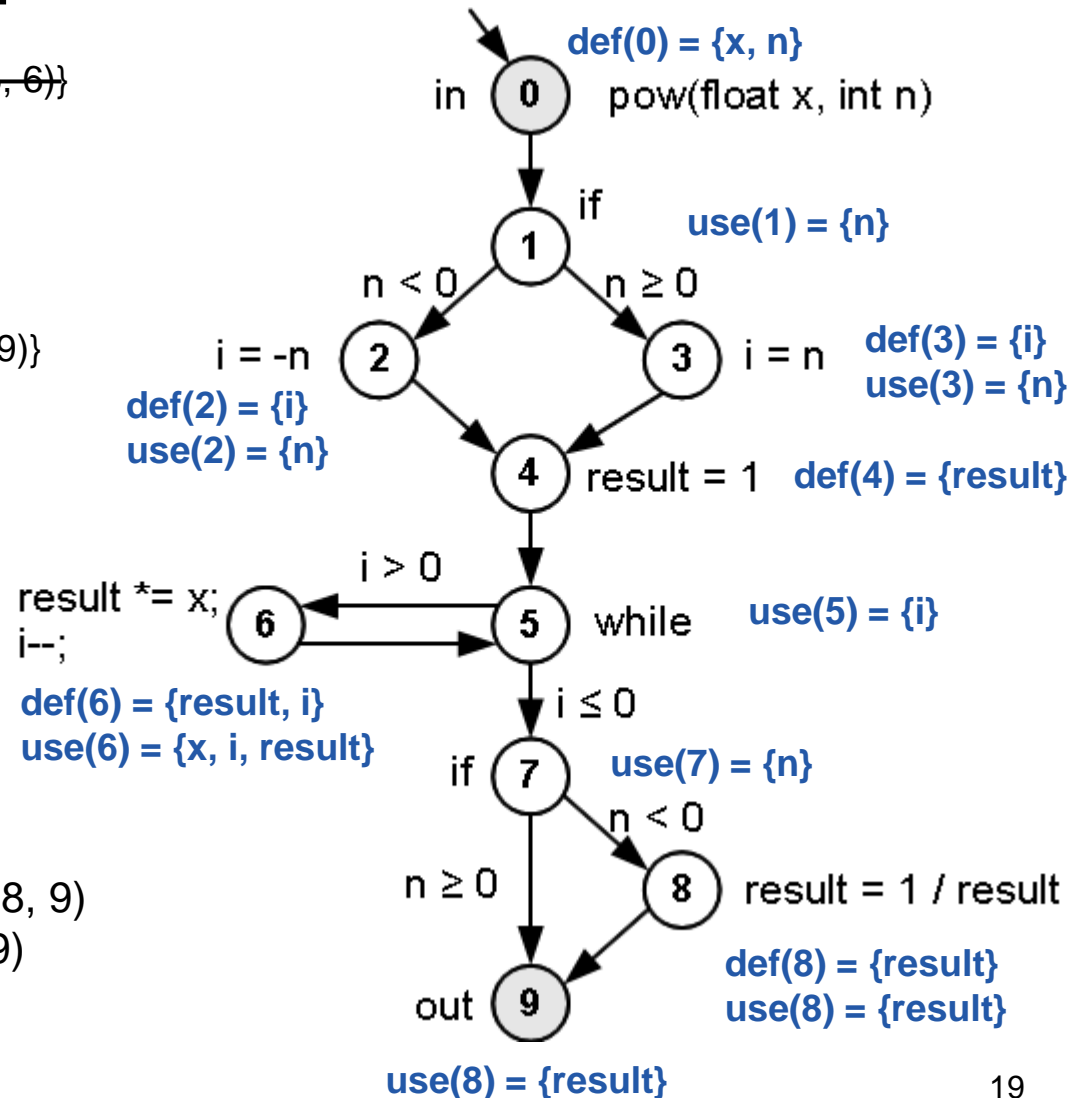
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\overline{0, 1, 2, 4, 5, 6}), (\overline{0, 1, 3, 4, 5, 6})\}$   
 $du(0, n) = \{(\overline{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\overline{2, 4, 5})\}$   
 $du(3, i) = \{(\overline{3, 4, 5})\}$   
 $du(4, result) = \{(\overline{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\overline{6, 5})\}$   
 $du(6, result) = \{(\overline{6, 5, 6}), (\overline{6, 5, 7, 8}), (6, 5, 7, 9)\}$   
 $du(8, result) = \{(\overline{8, 9})\}$

- All-Definitions:

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$







# Defs-Uses-Überdeckungskriterien

## All-Definitions – Beispiel

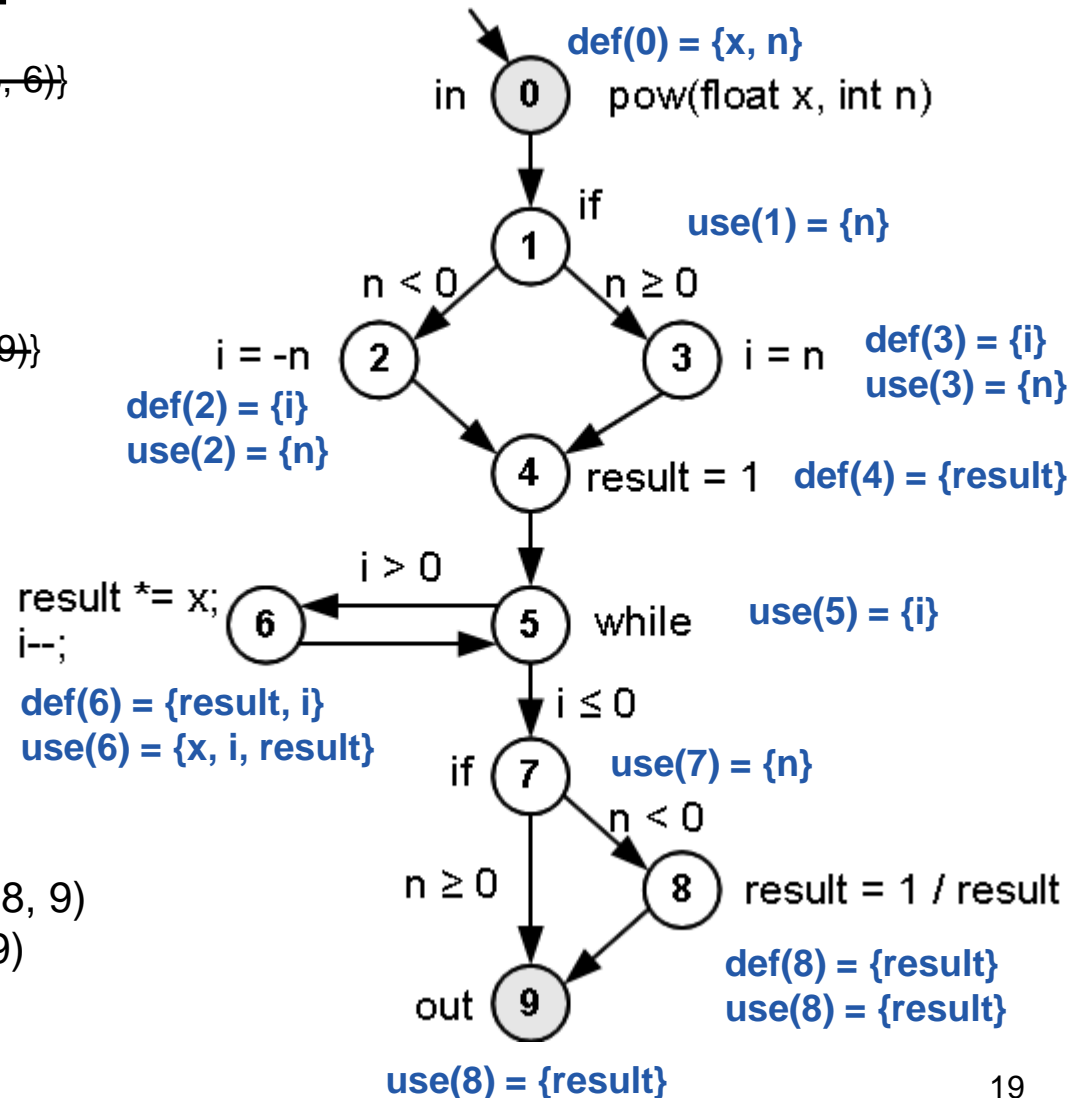
- Alle Pfadmenge  $du(n, v)$ :

$du(0, x) = \{(\cancel{0, 1, 2, 4, 5, 6}), (\cancel{0, 1, 3, 4, 5, 6})\}$   
 $du(0, n) = \{(\cancel{0, 1}), (0, 1, 3, 4, 5, 7)\}$   
 $du(2, i) = \{(\cancel{2, 4, 5})\}$   
 $du(3, i) = \{(\cancel{3, 4, 5})\}$   
 $du(4, result) = \{(\cancel{4, 5, 6}), (4, 5, 7, 9)\}$   
 $du(6, i) = \{(\cancel{6, 5})\}$   
 $du(6, result) = \{(\cancel{6, 5, 6}), (\cancel{6, 5, 7, 8}), (\cancel{6, 5, 7, 9})\}$   
 $du(8, result) = \{(\cancel{8, 9})\}$

- All-Definitions:**

– Überdeckung durch folgende Tests

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Uses

---

- Überdeckungskriterium: **All-Uses**
  - Für alle Definitionen einer Variable, überdecke mindestens einen bezüglich dieser Variablen definitionsfreien Pfad zu jedem lesenden Zugriff (c-use oder p-use) dieser Variablen



# Defs-Uses-Überdeckungskriterien

## All-Uses

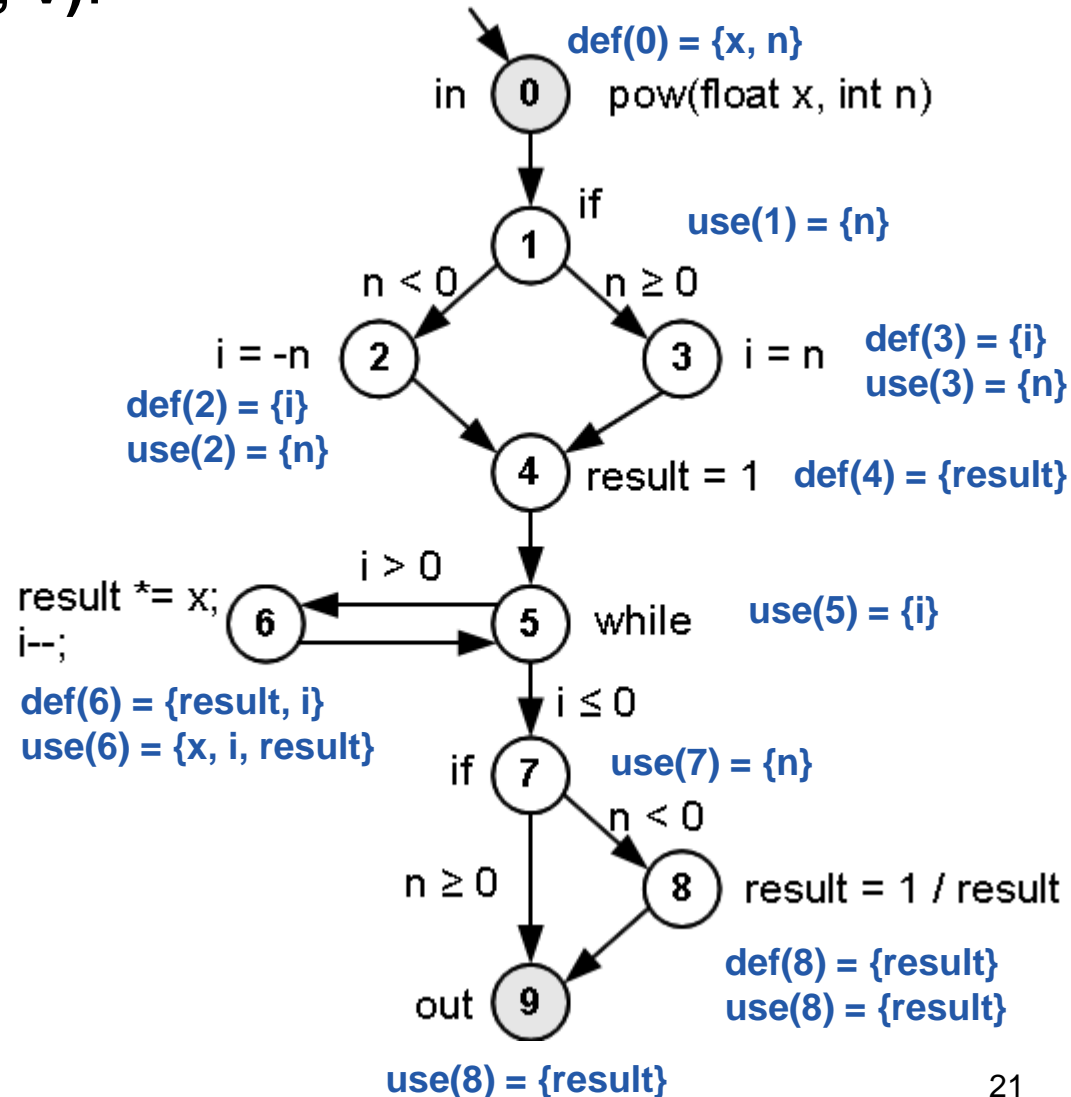
---

- Überdeckungskriterium: **All-Uses**
  - **Für alle Definitionen einer Variable**, überdecke **mindestens einen** bezüglich dieser Variablen definitionsfreien Pfad zu jedem lesenden Zugriff (c-use oder p-use) dieser Variablen
  - Etwas formaler: **Für jede Pfadmenge  $du(n, n', v)$**  muss **mindestens ein Pfad** aus dieser Menge überdeckt werden

# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

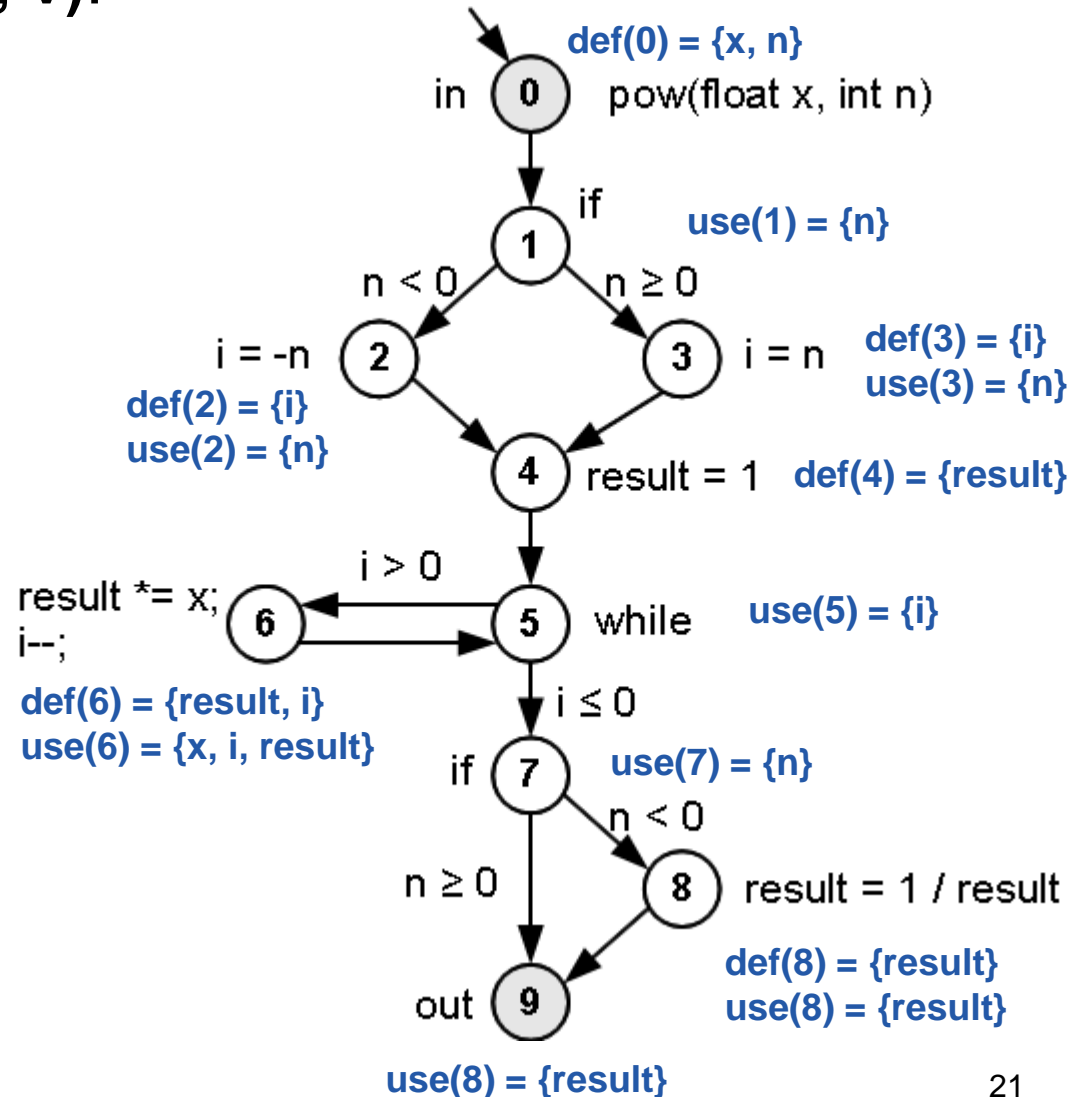




# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :  
 $du(0, 6, x)$



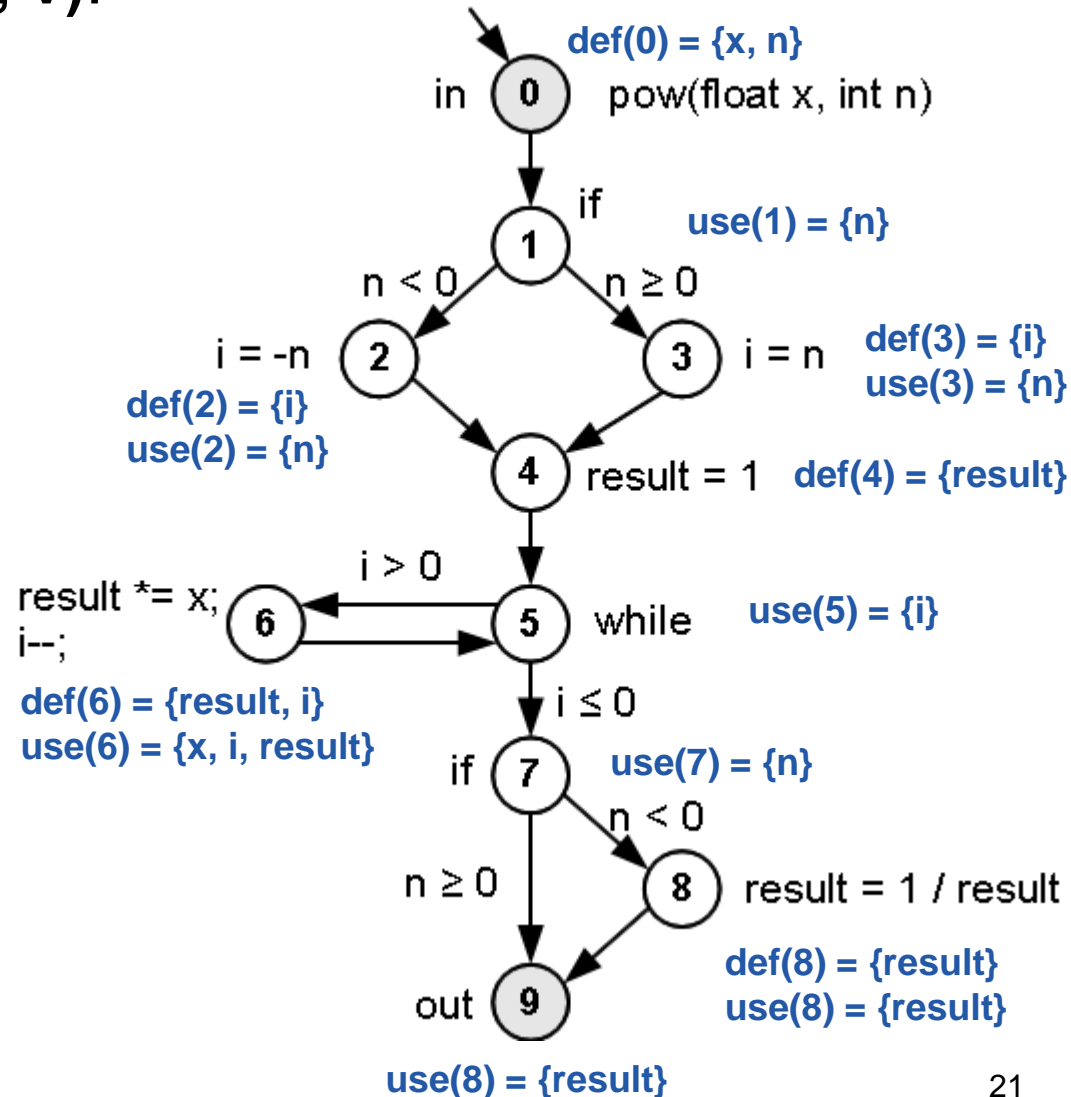


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$



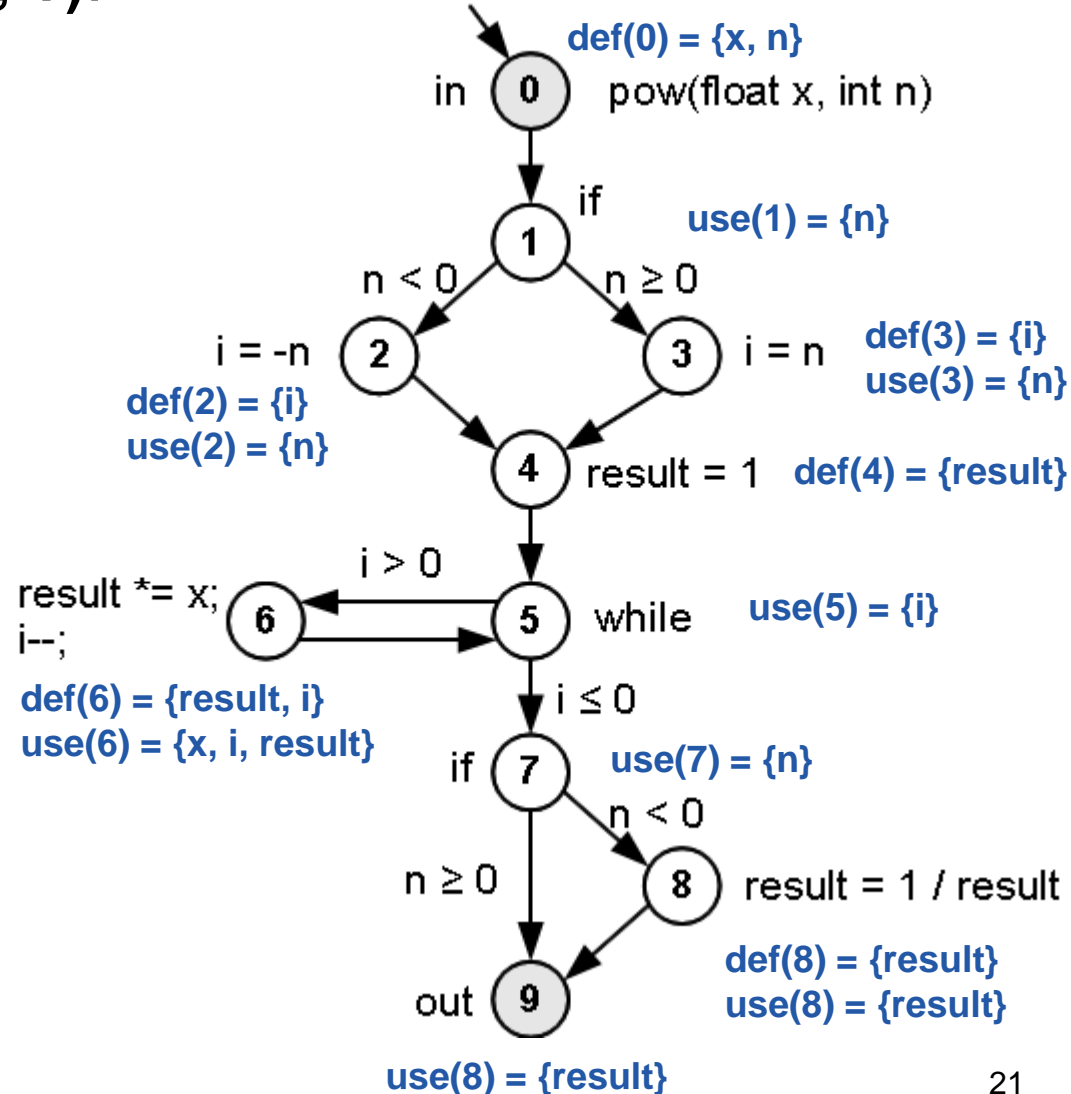


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$



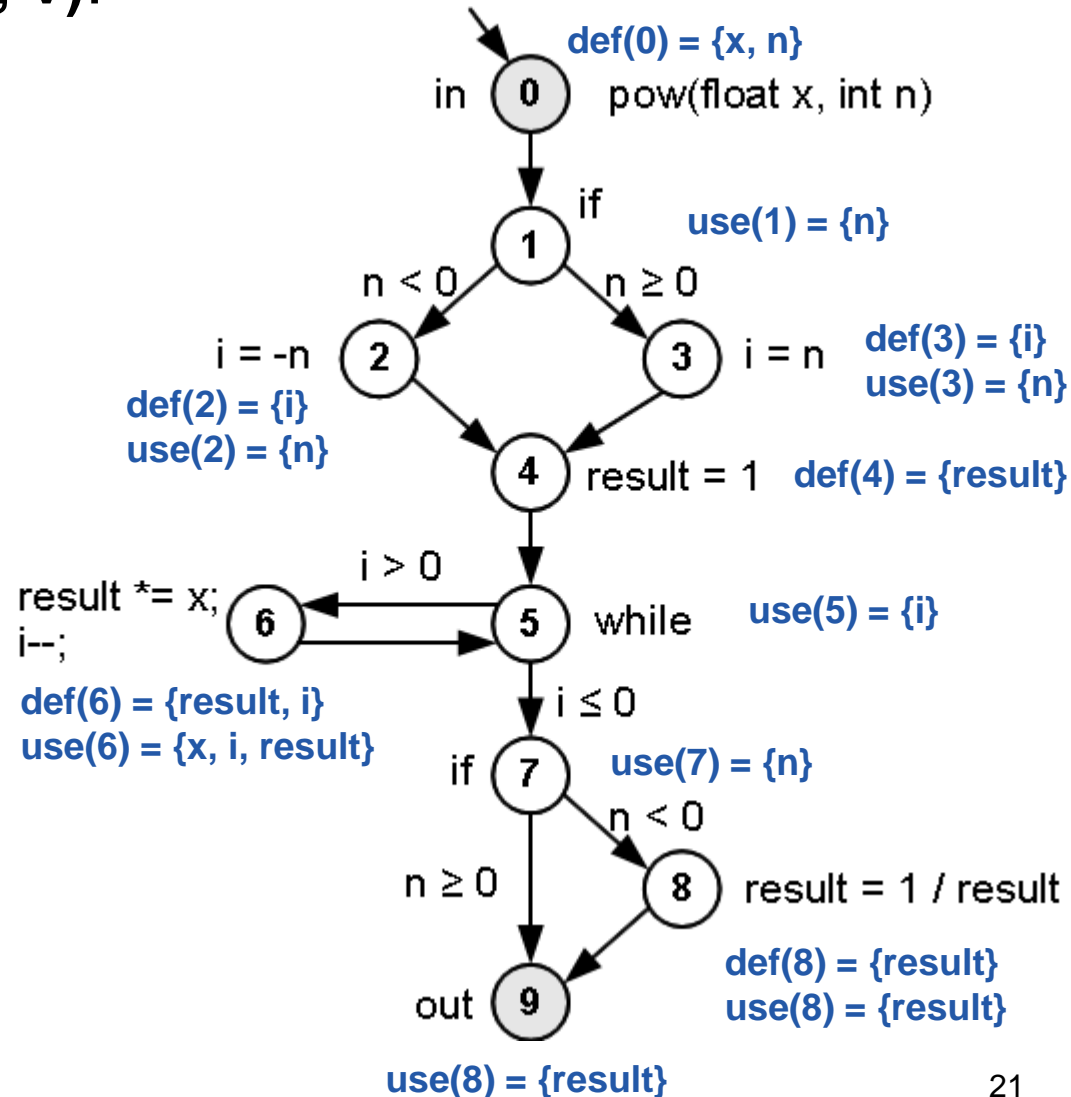


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$





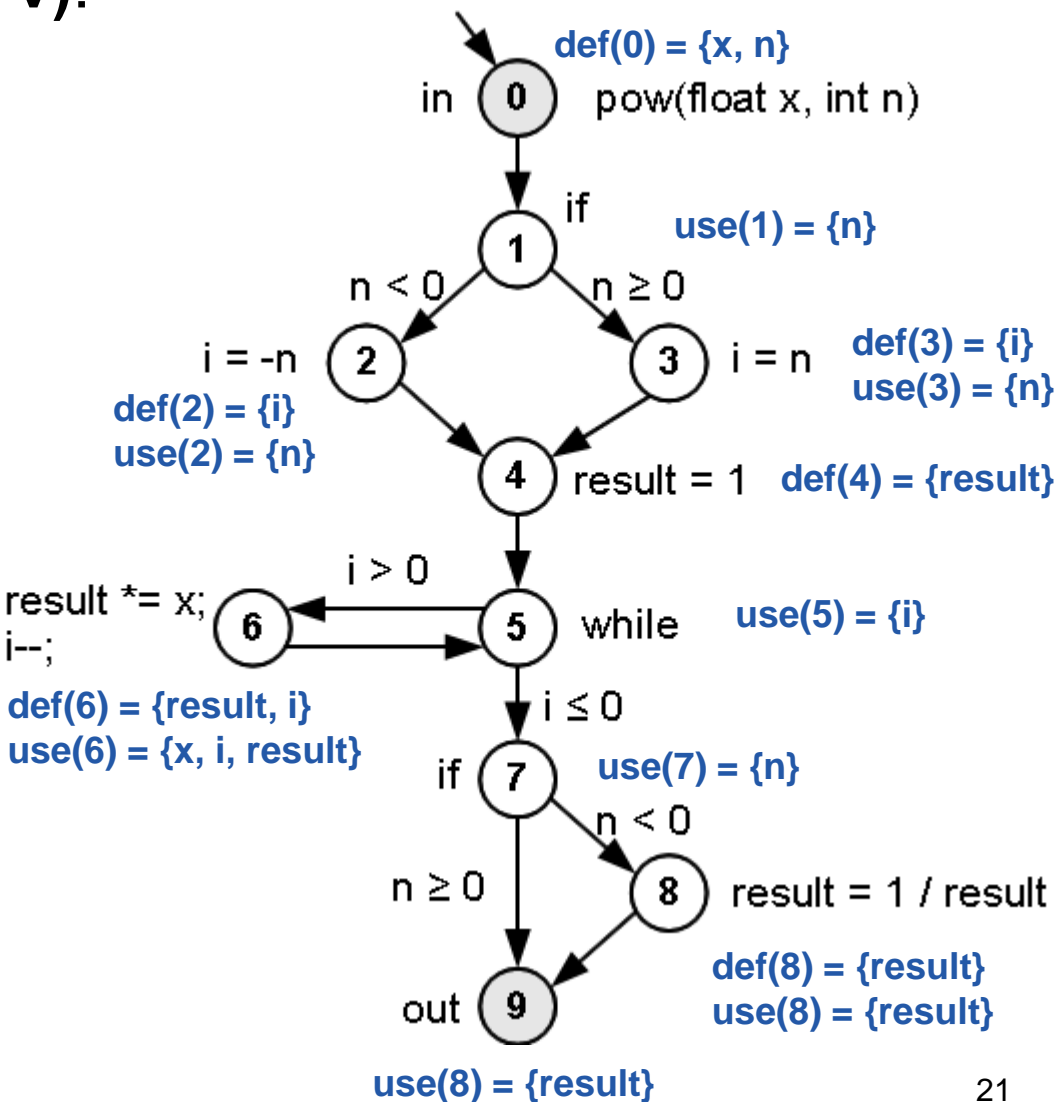


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$



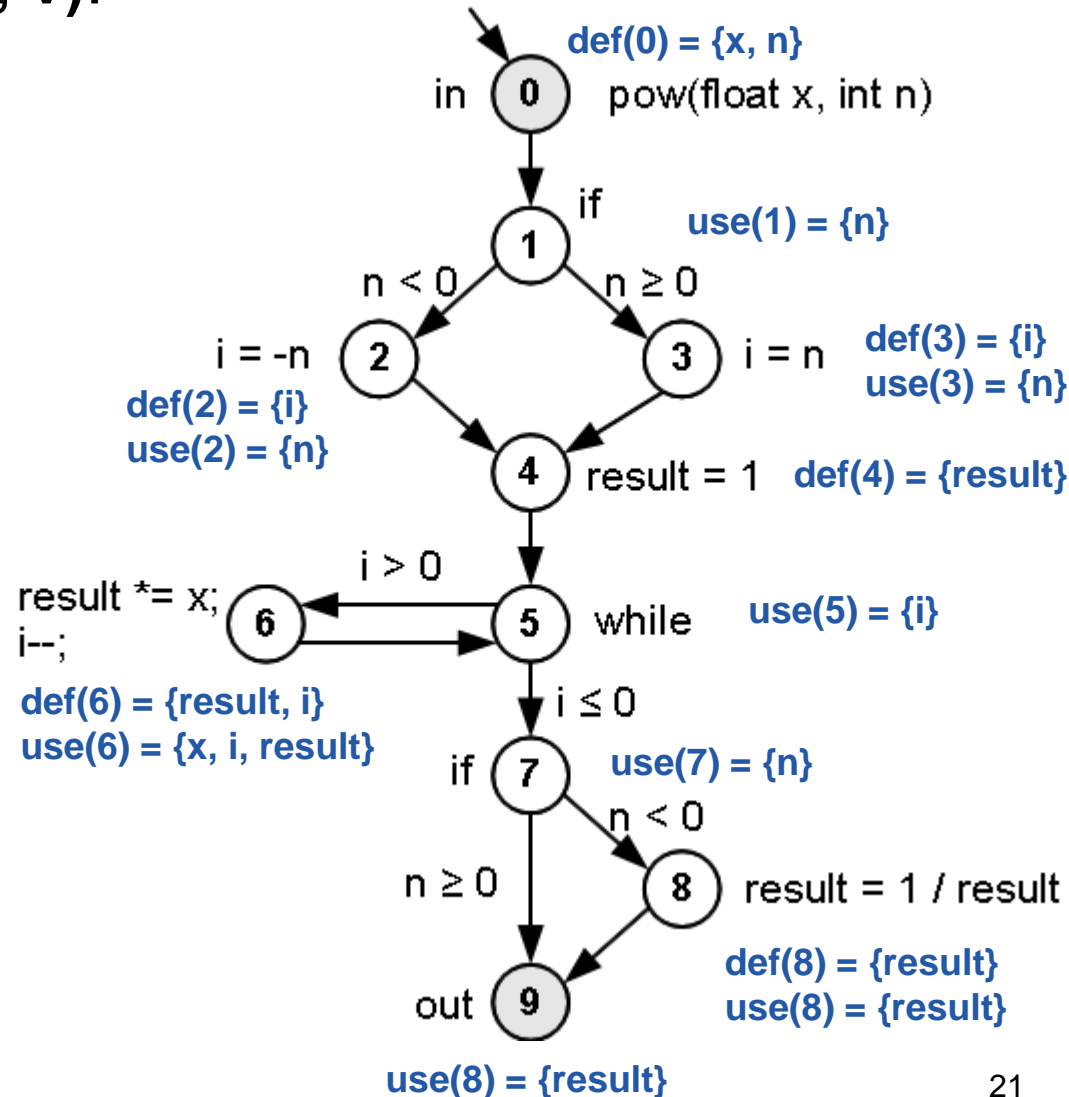


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$



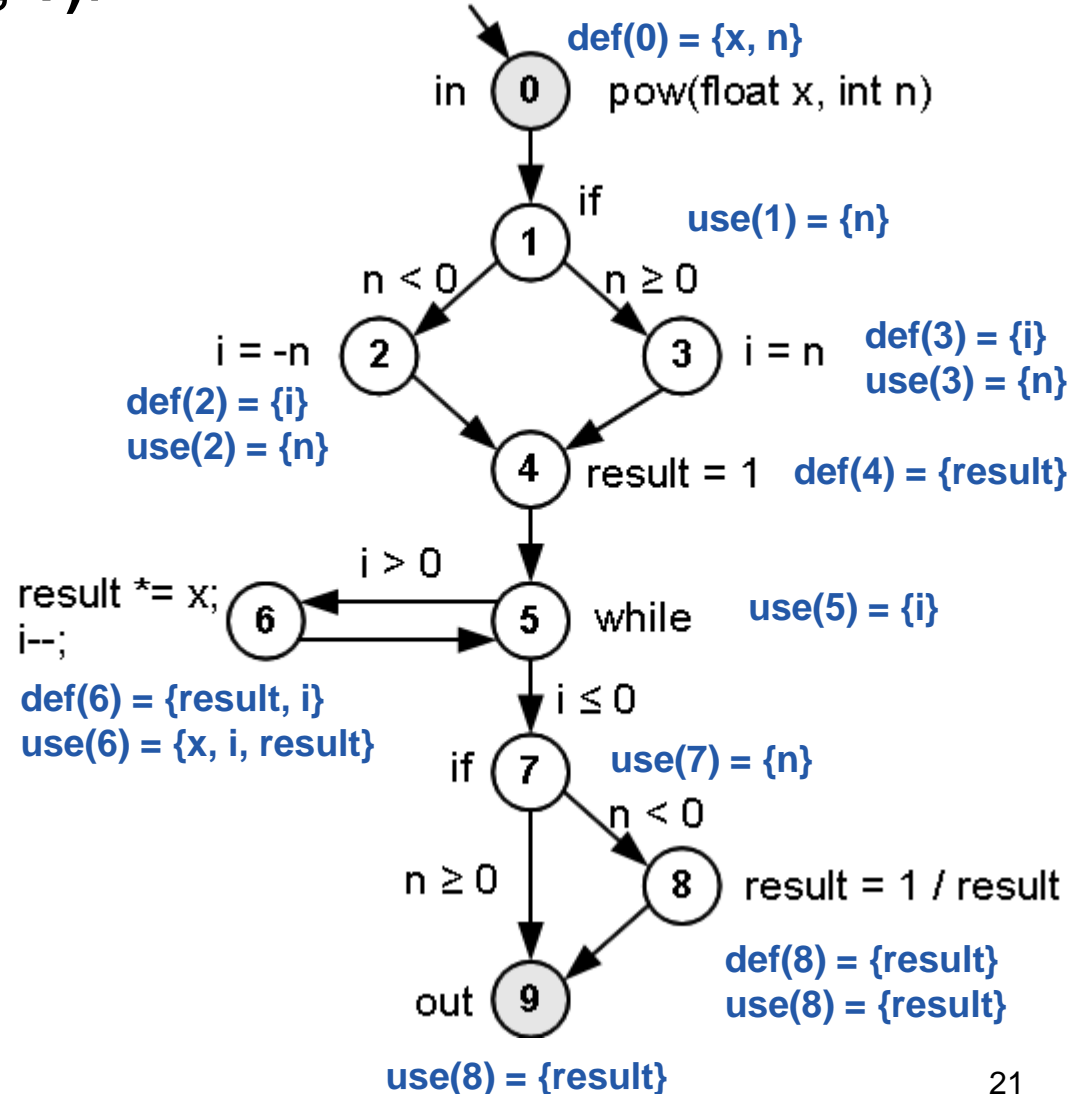


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$





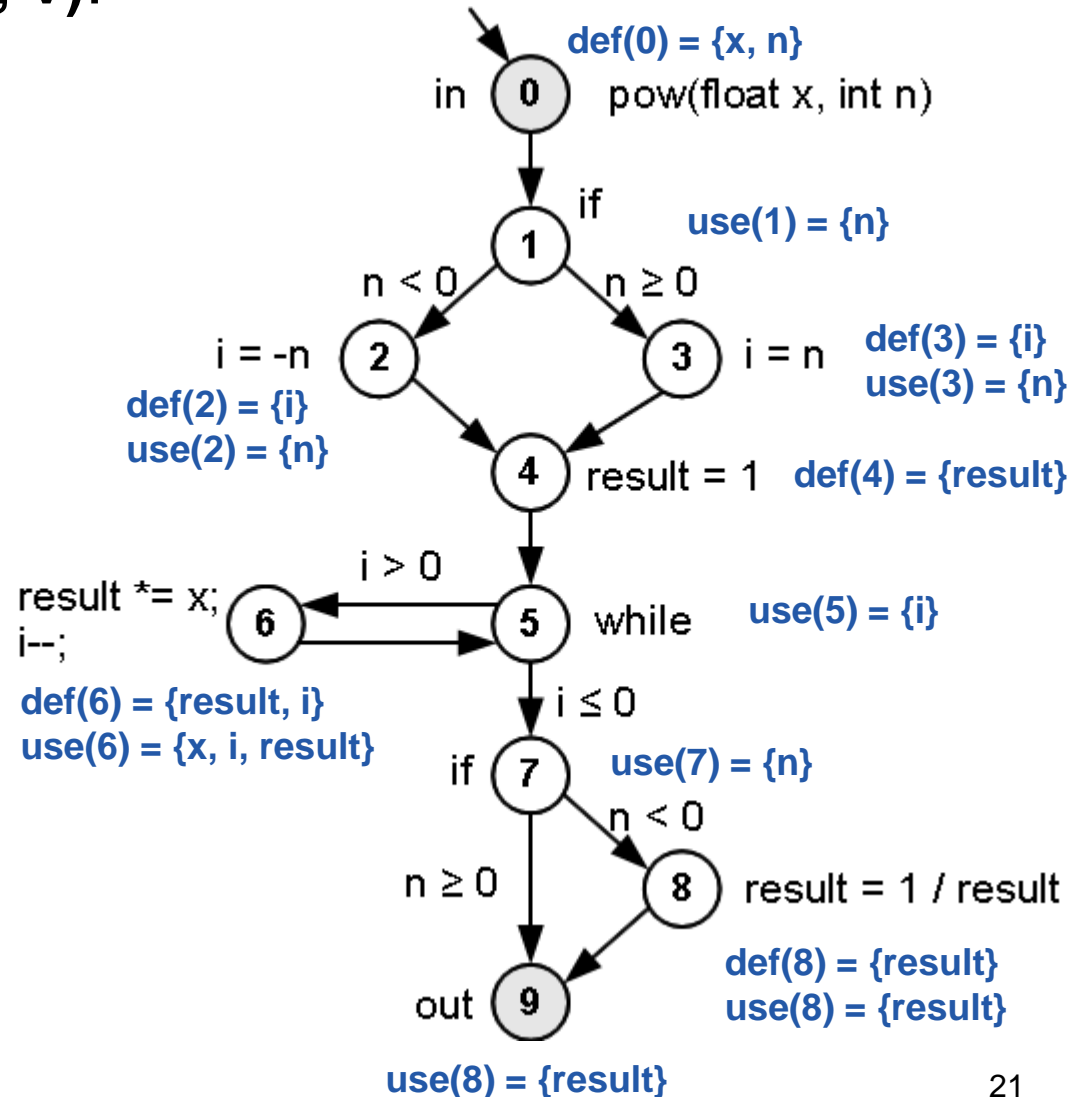
# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
 $= \{(0, 1)\}$   
 $du(2, 5, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, 5, i)$   
 $= \{(3, 4, 5)\}$

$du(0, 7, n) =$   
 $\{(0, 1, 3, 4, 5, 7)\}$



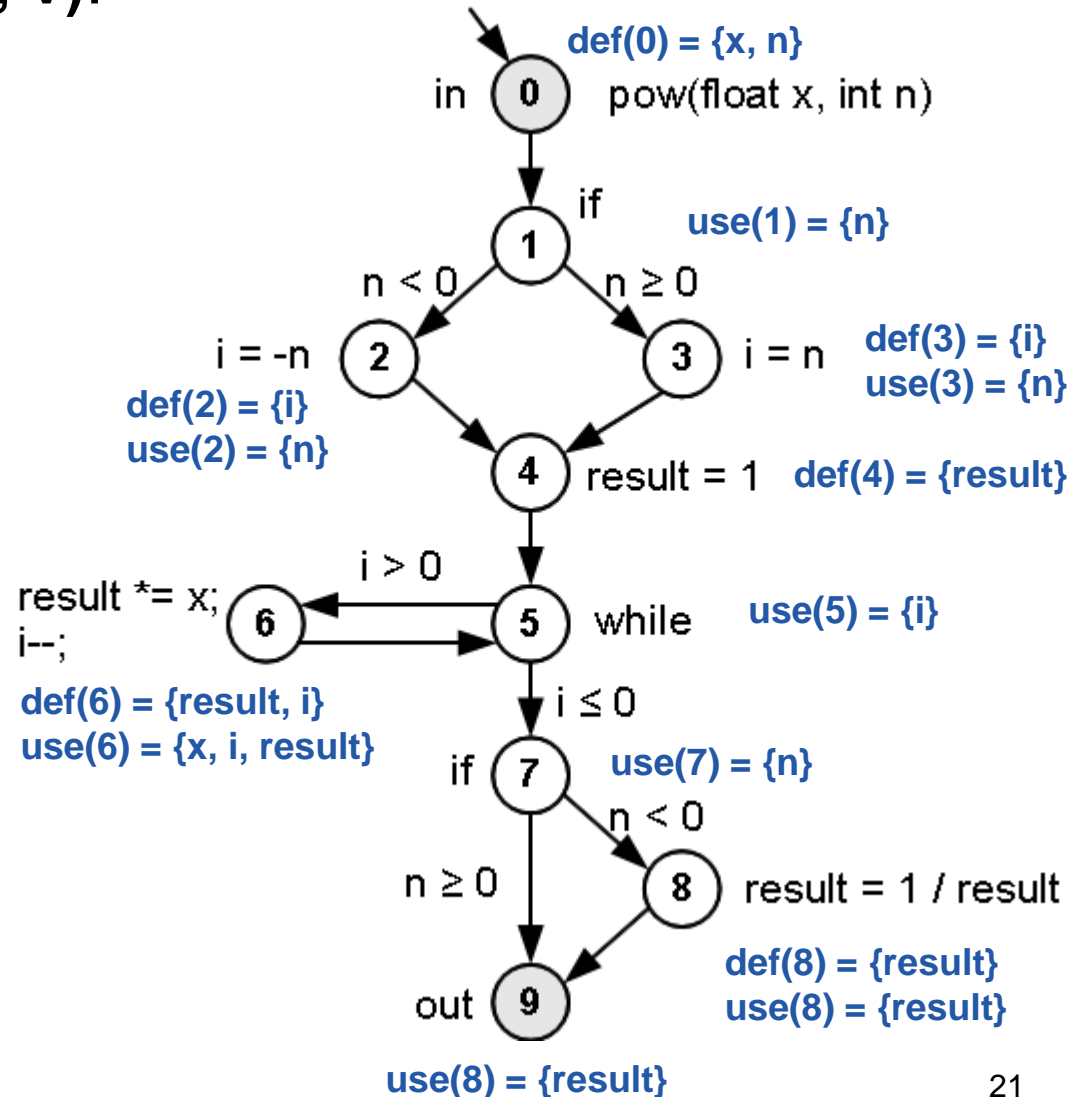


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$



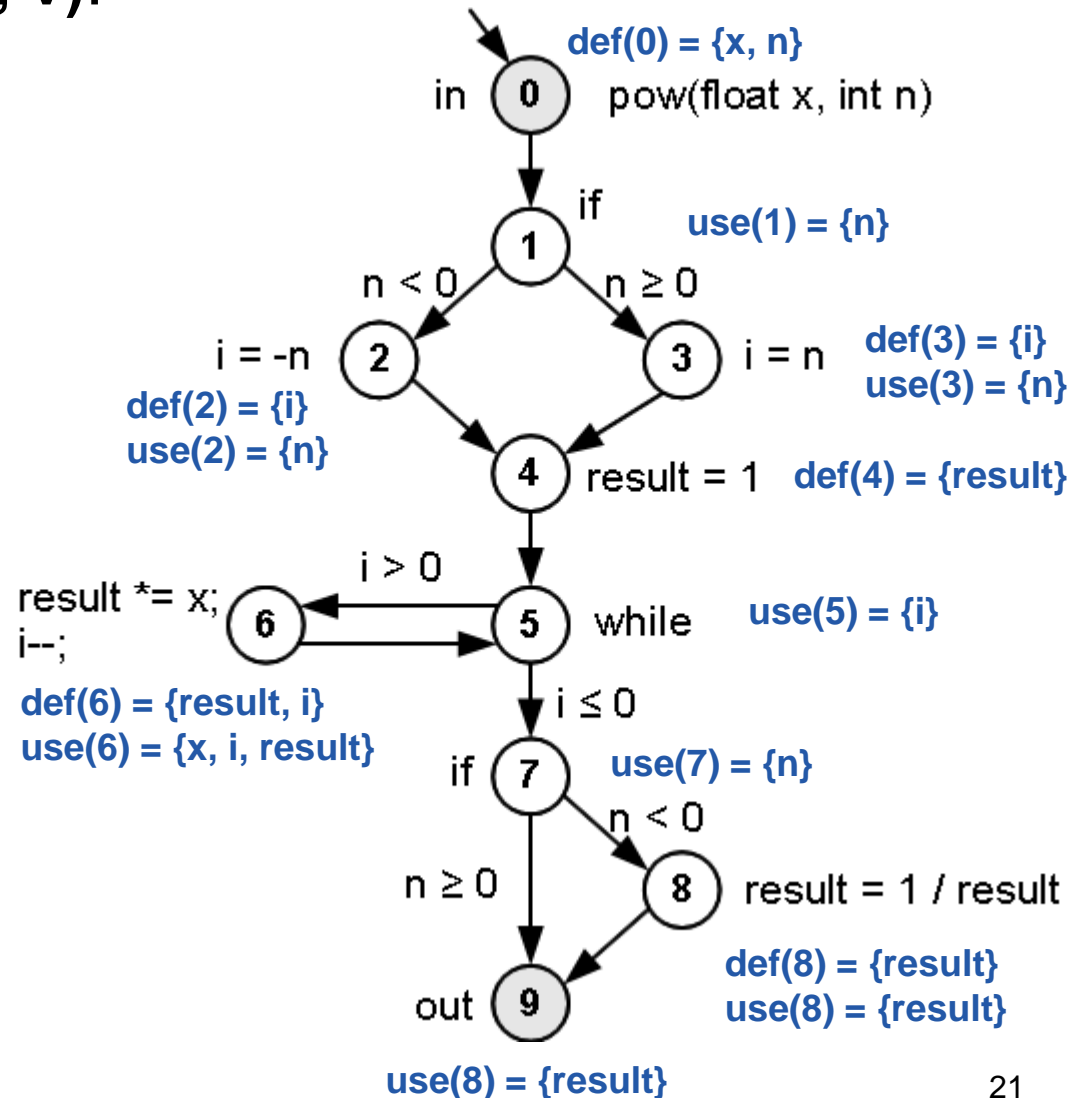


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(0, 7, n)$   
=  $\{(0, 1, 3, 4, 5, 7)\}$



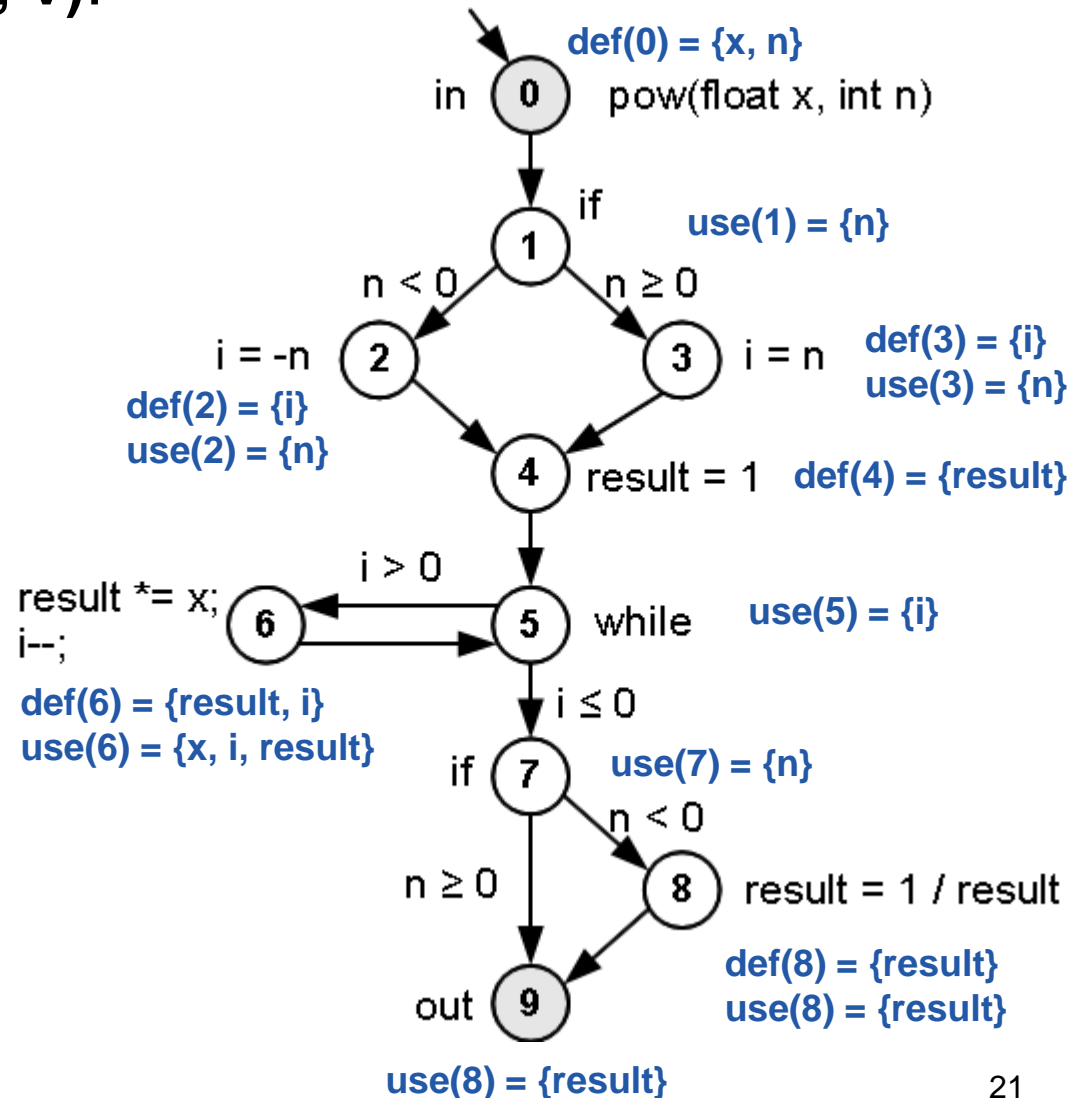


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$



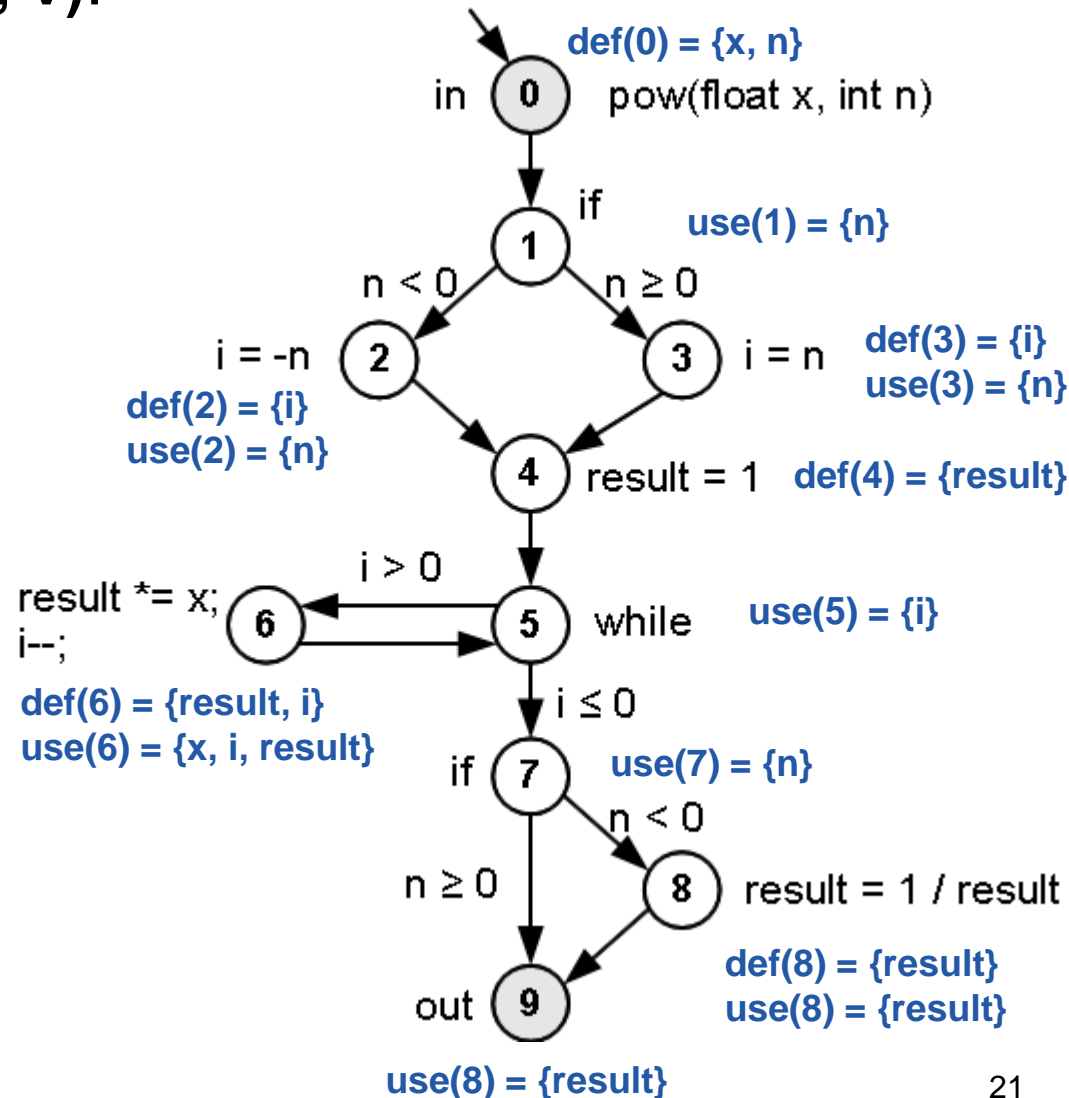


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, \text{result})$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, \text{result})$   
=  $\{(4, 5, 7, 8)\}$





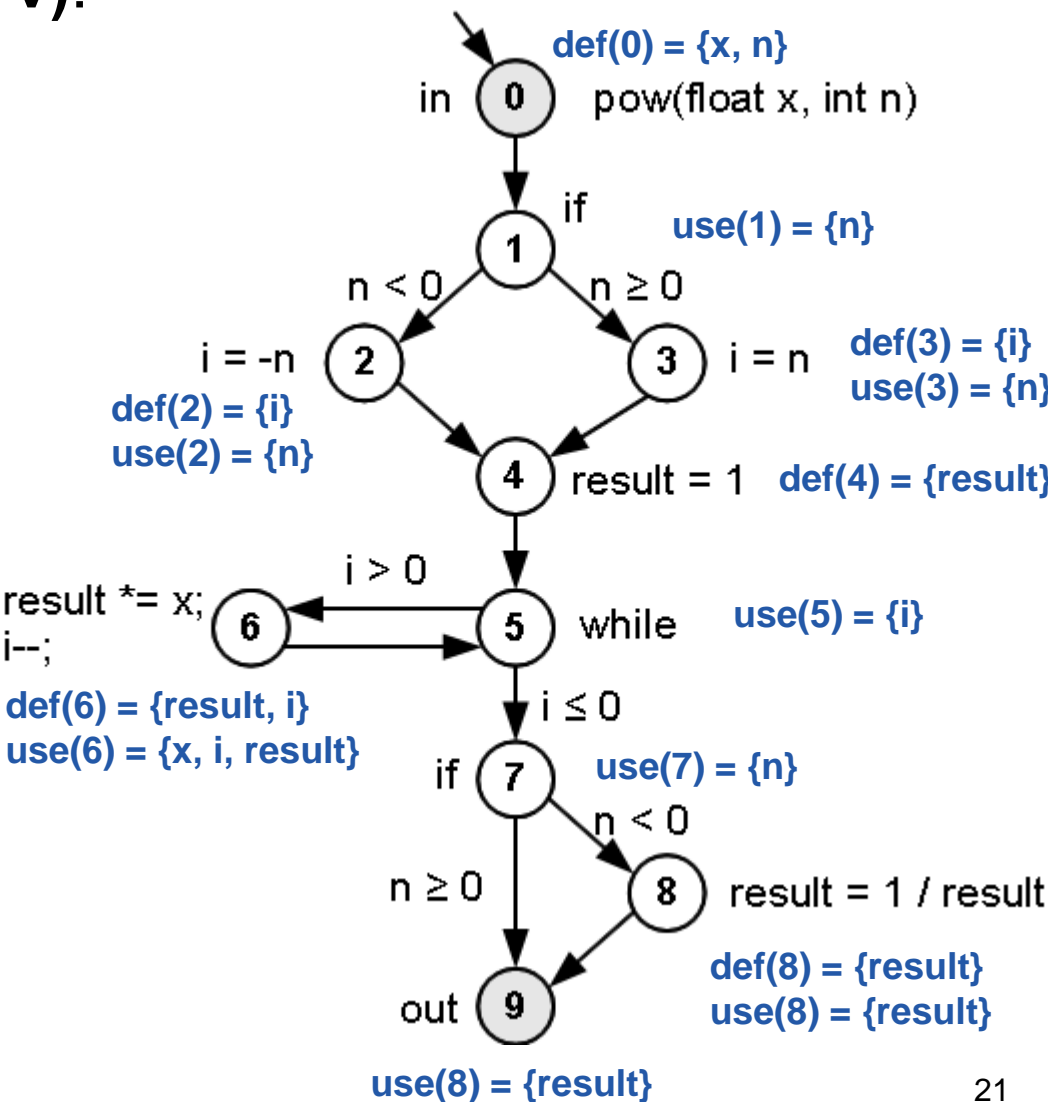


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$

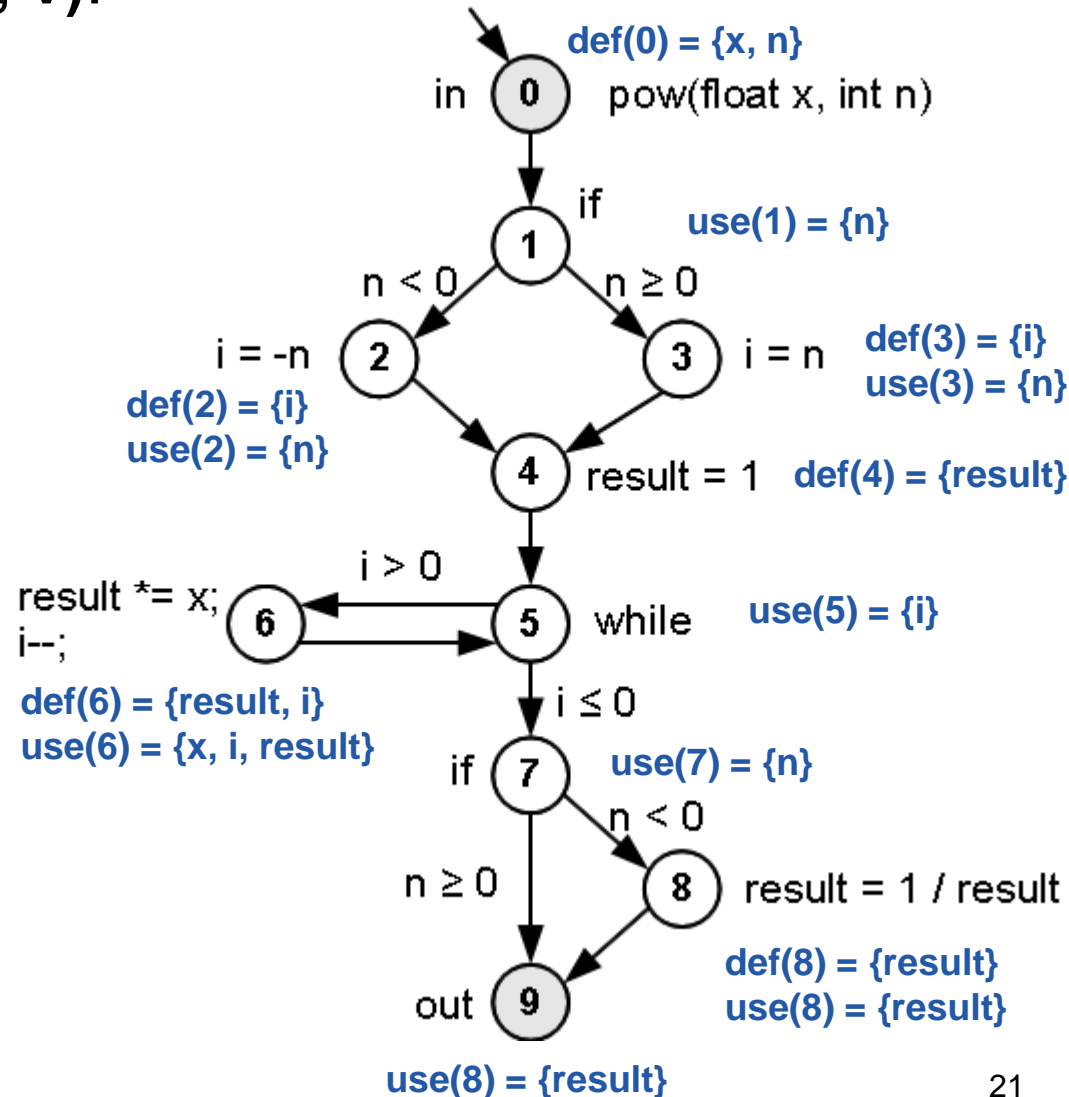


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
 $= \{(0, 1)\}$   
 $du(2, 5, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, 5, i)$   
 $= \{(3, 4, 5)\}$   
 $du(4, 6, result)$   
 $= \{(4, 5, 6)\}$   
 $du(4, 8, result)$   
 $= \{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
 $= \{(4, 5, 7, 9)\}$



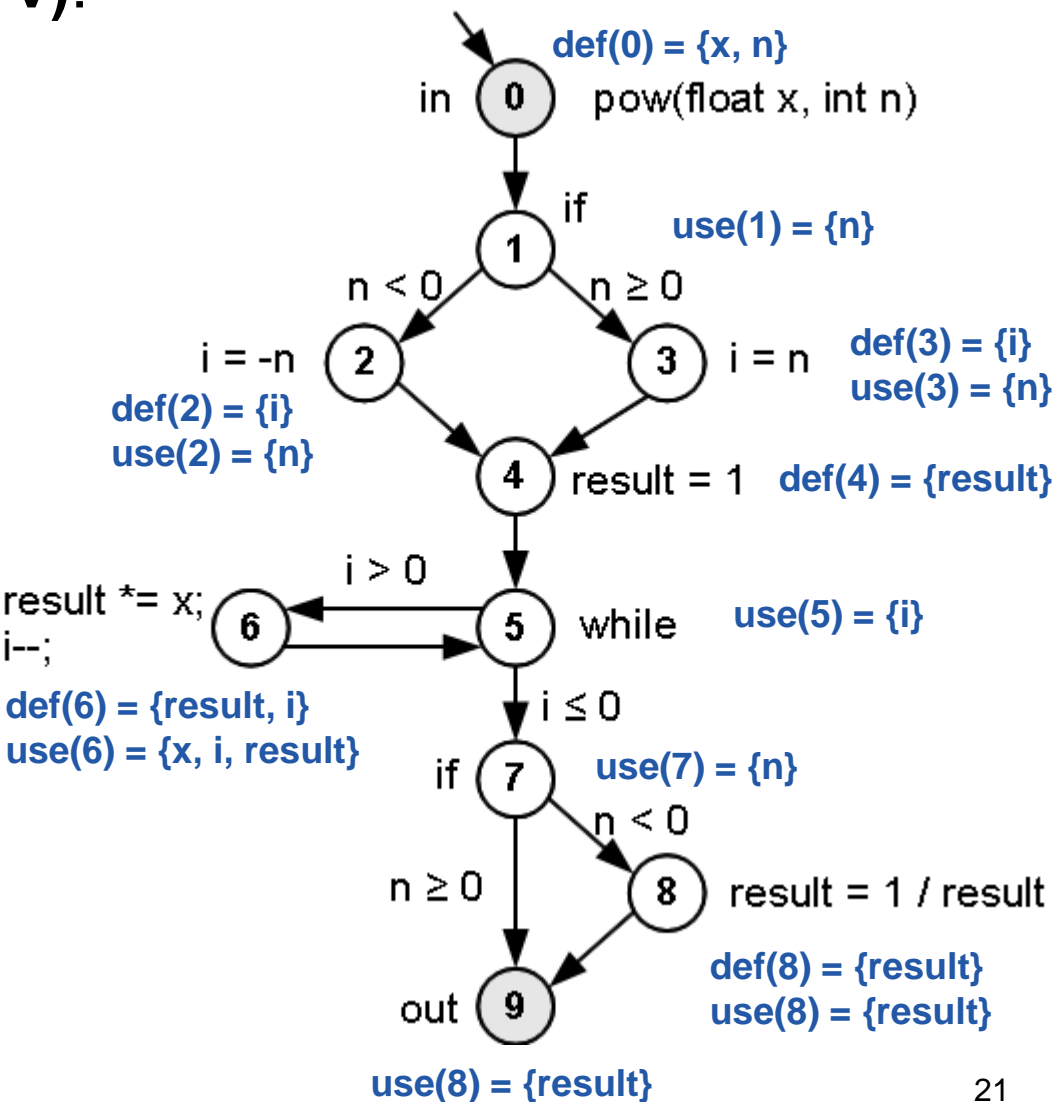


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$



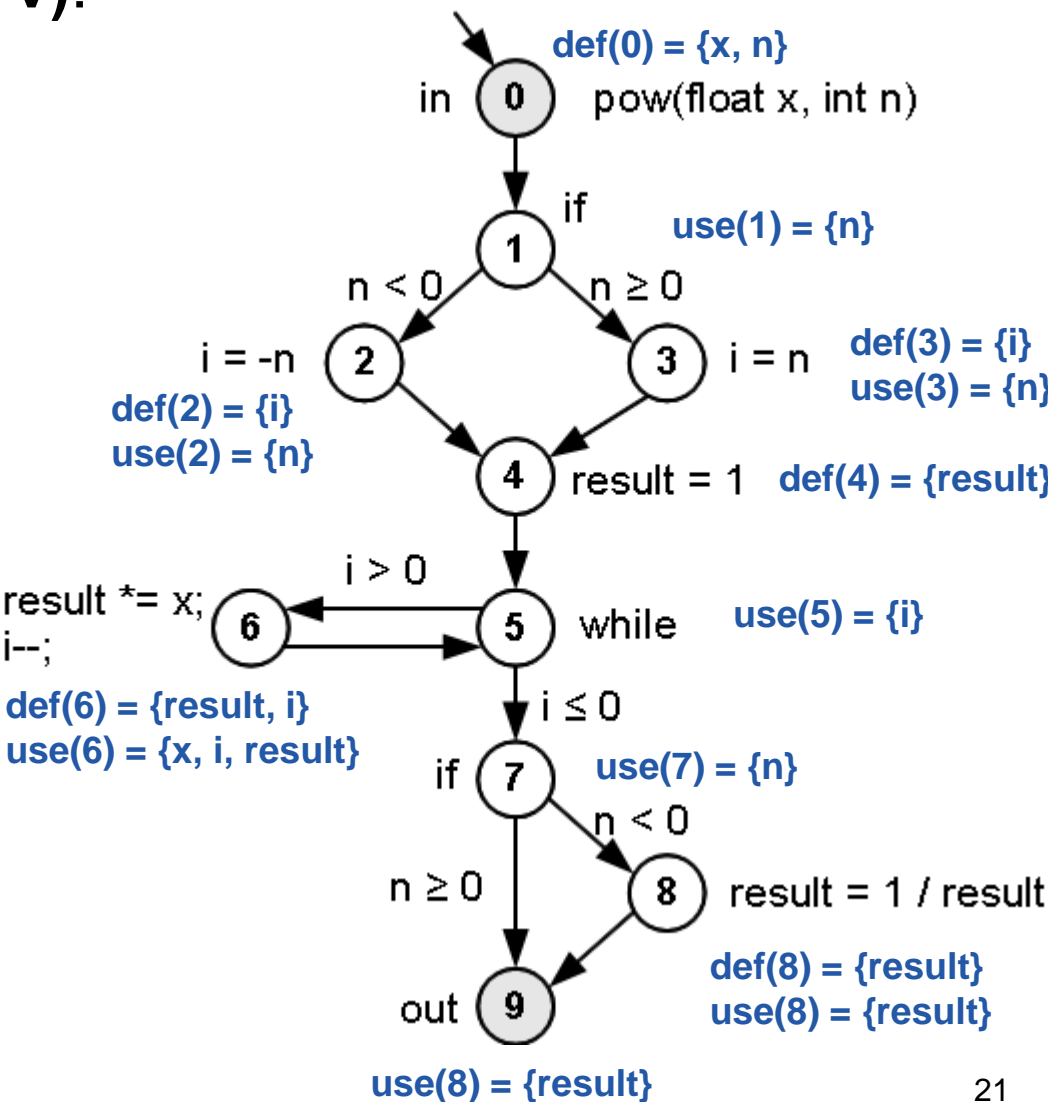


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(0, 7, n) = \{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$



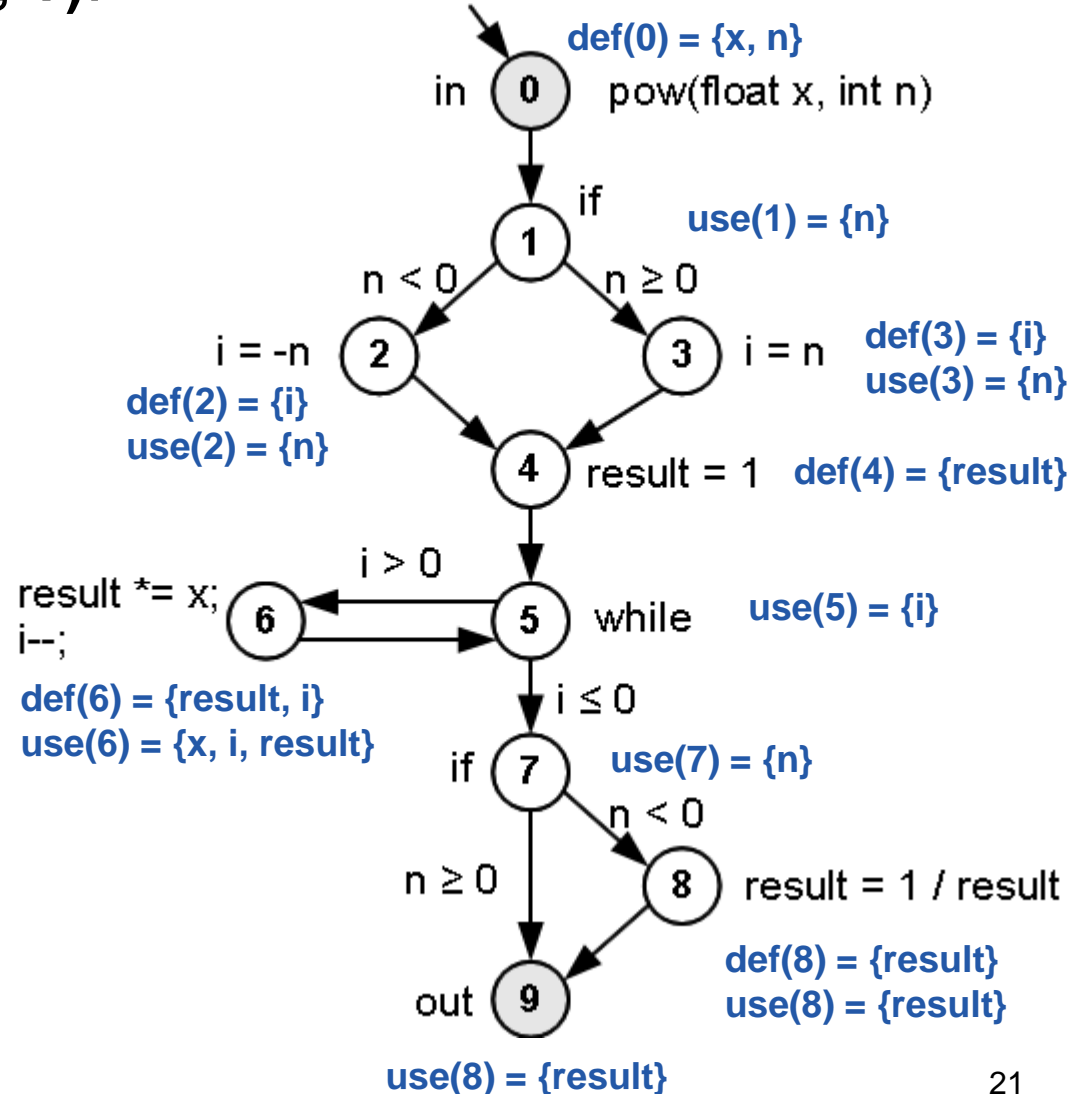


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
 $= \{(0, 1)\}$   
 $du(0, 7, n) = \{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, 5, i)$   
 $= \{(3, 4, 5)\}$   
 $du(4, 6, result)$   
 $= \{(4, 5, 6)\}$   
 $du(4, 8, result)$   
 $= \{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
 $= \{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
 $= \{(6, 5)\}$   
 $du(6, 6, result)$



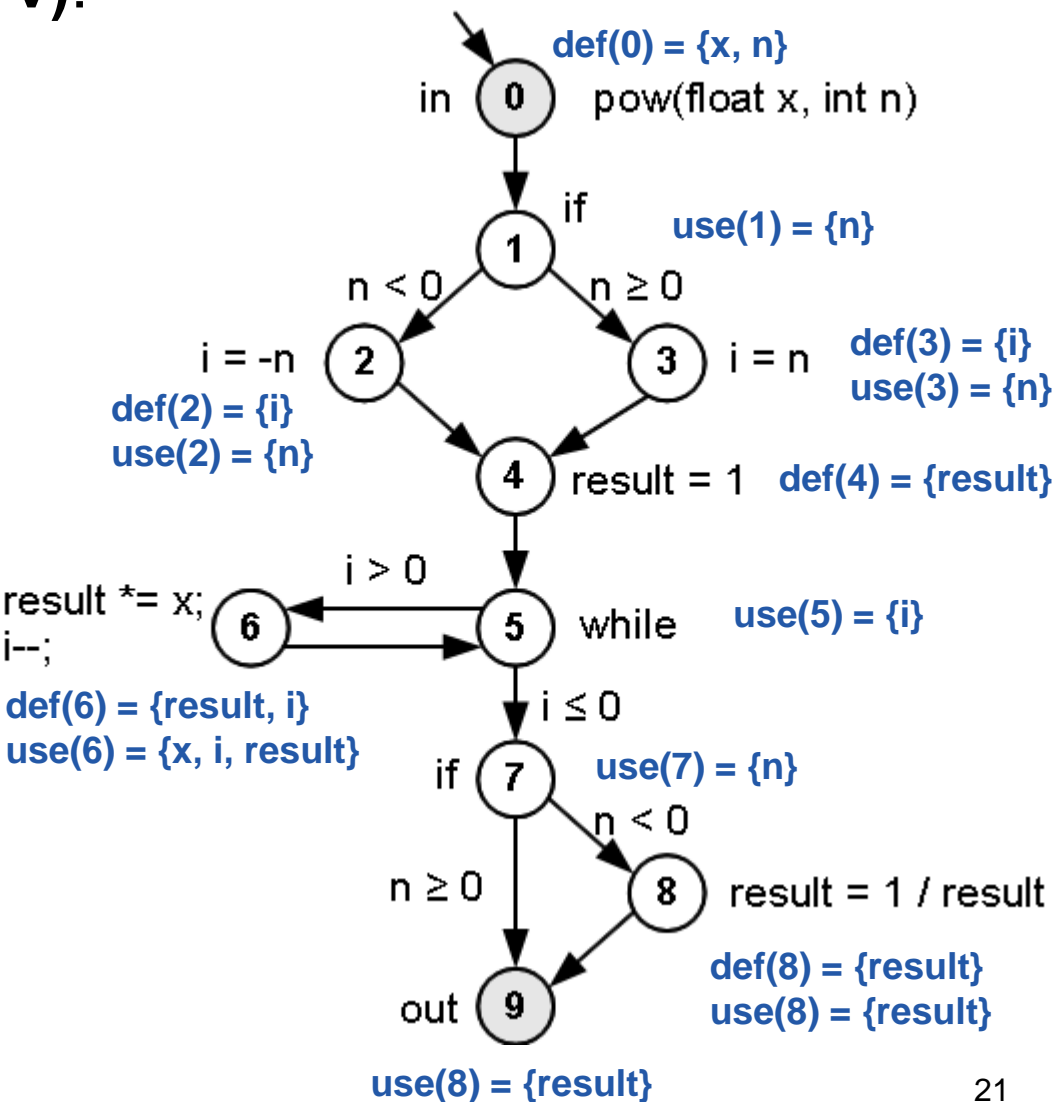


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$



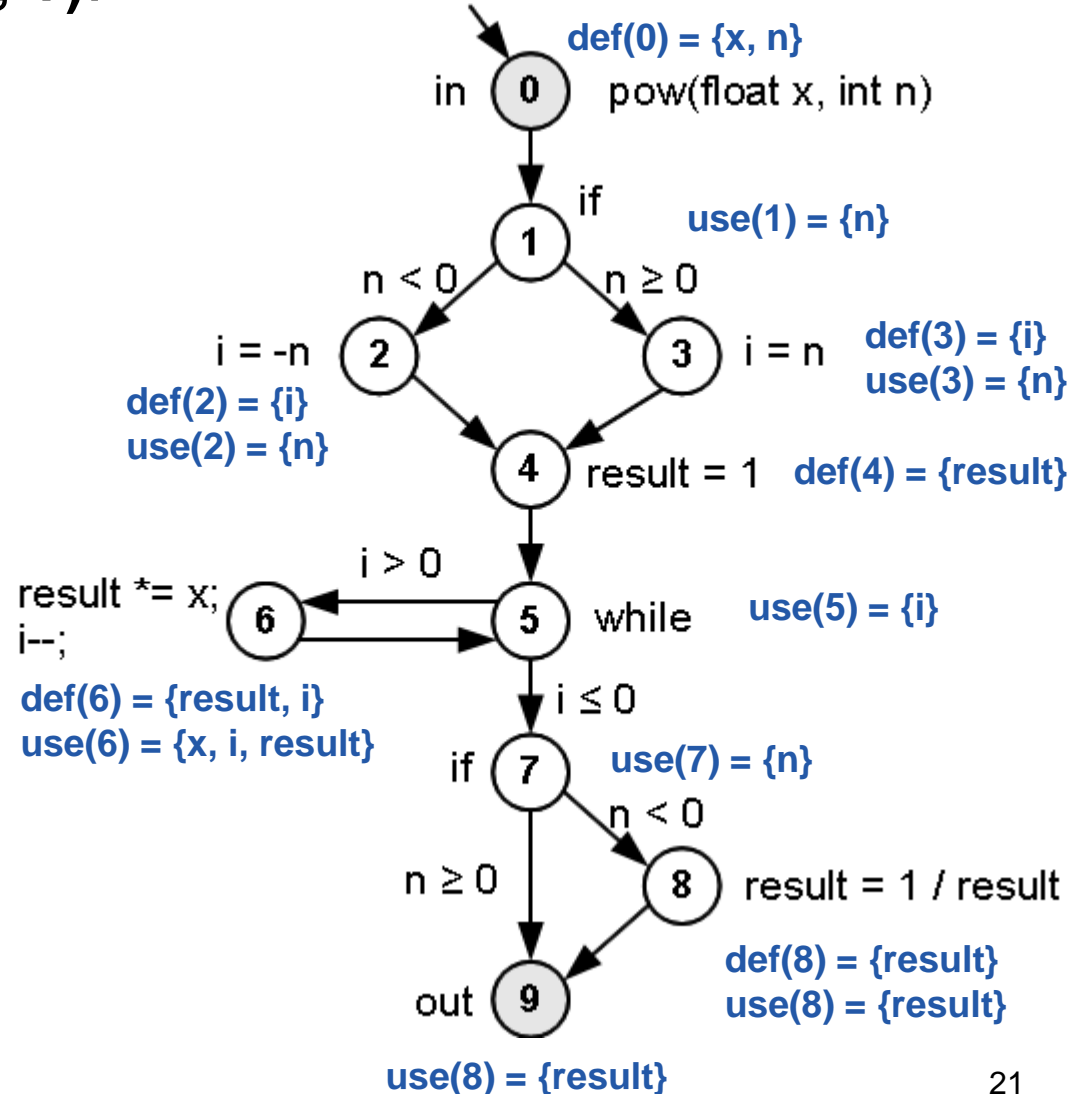


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$   
 $du(6, 8, result)$



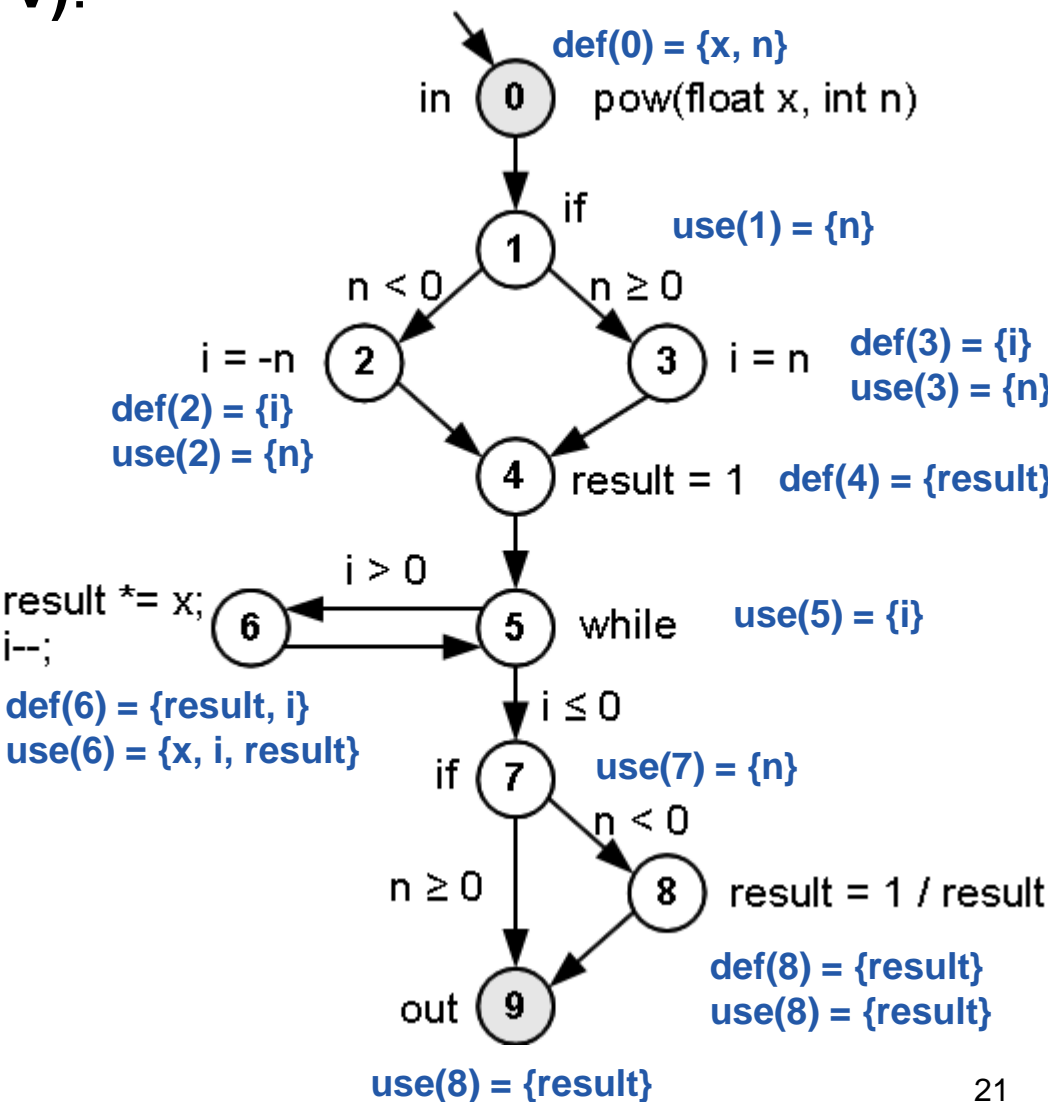


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$        $du(0, 7, n) =$   
=  $\{(0, 1)\}$        $\{(0, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$   
 $du(6, 8, result)$   
=  $\{(6, 5, 7, 8)\}$





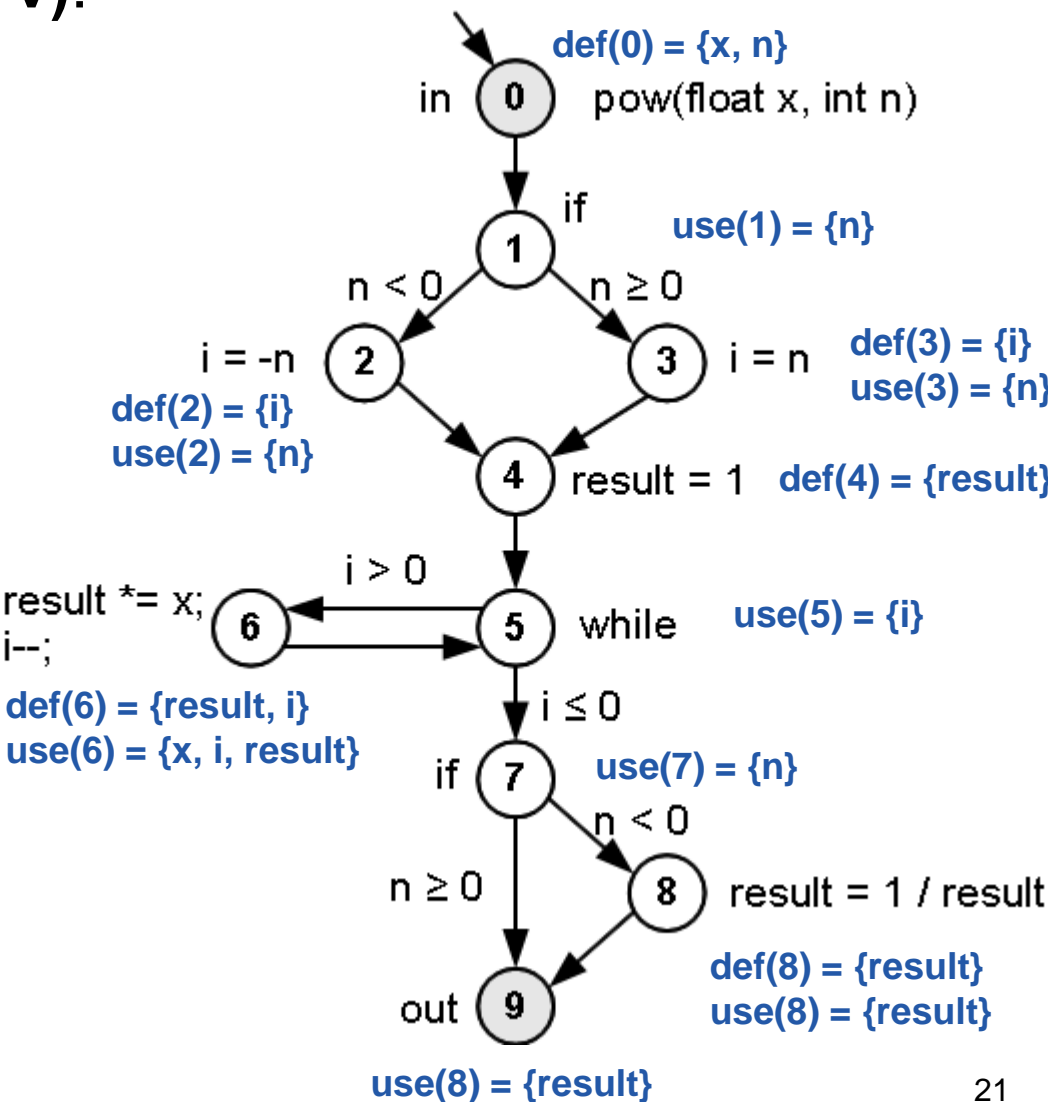


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$   
 $du(6, 8, result)$   
=  $\{(6, 5, 7, 8)\}$   
 $du(6, 9, result)$

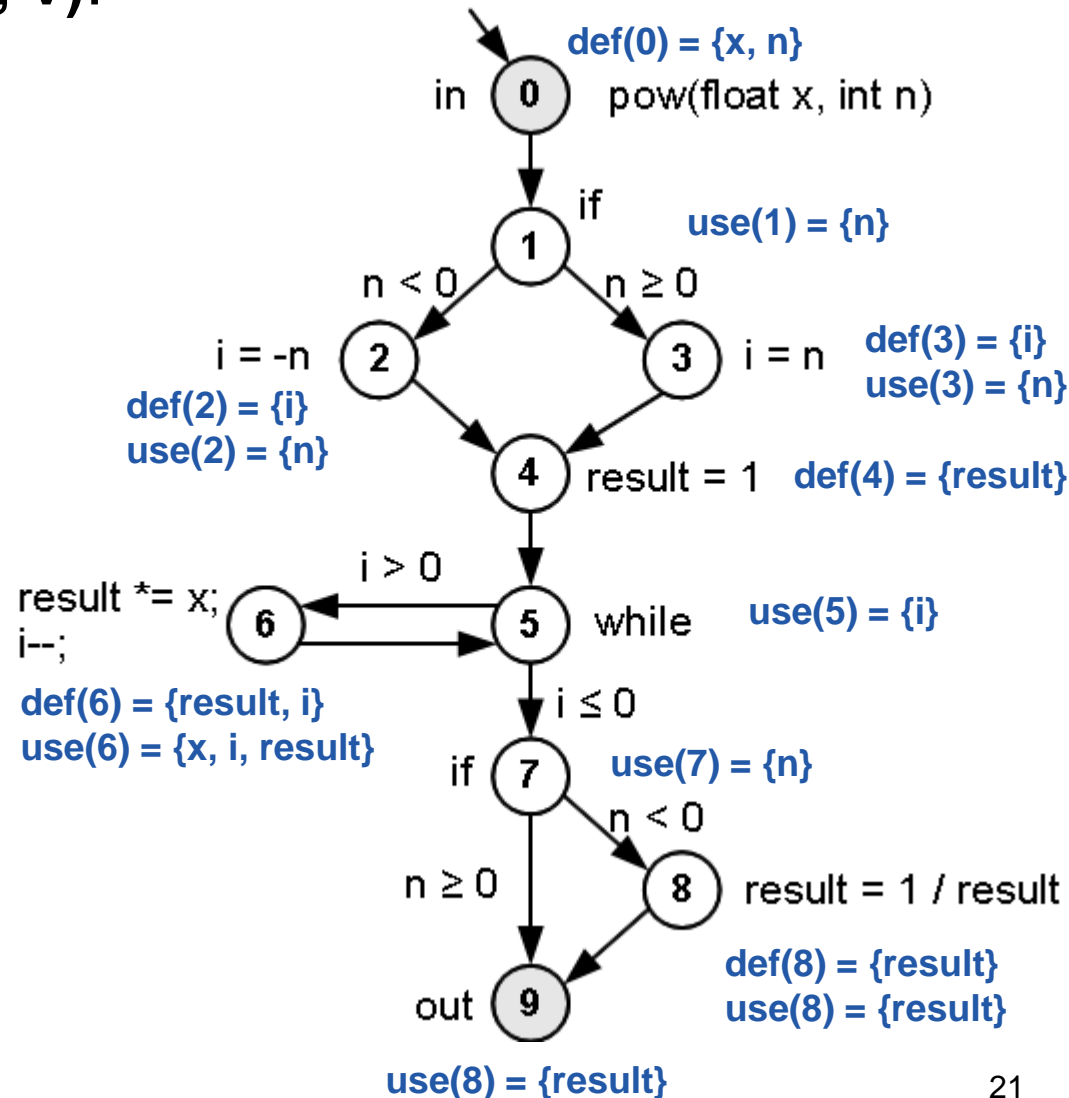


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
 $= \{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
 $= \{(0, 1)\}$   
 $du(2, 5, i)$   
 $= \{(2, 4, 5)\}$   
 $du(3, 5, i)$   
 $= \{(3, 4, 5)\}$   
 $du(4, 6, result)$   
 $= \{(4, 5, 6)\}$   
 $du(4, 8, result)$   
 $= \{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
 $= \{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
 $= \{(6, 5)\}$   
 $du(6, 6, result)$   
 $= \{(6, 5, 6)\}$   
 $du(6, 8, result)$   
 $= \{(6, 5, 7, 8)\}$   
 $du(6, 9, result)$   
 $= \{(6, 5, 7, 9)\}$



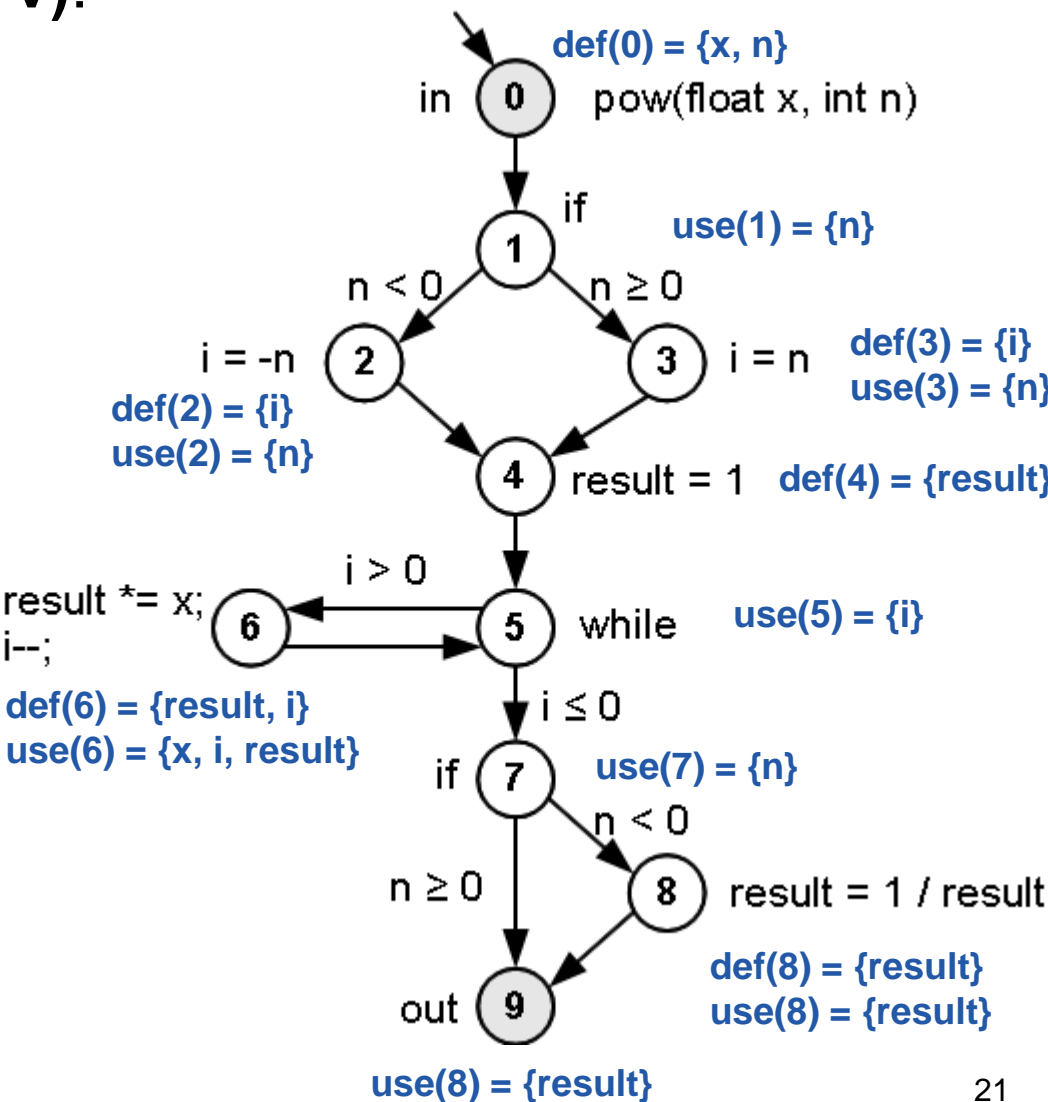


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$   
 $du(6, 8, result)$   
=  $\{(6, 5, 7, 8)\}$   
 $du(6, 9, result)$   
=  $\{(6, 5, 7, 9)\}$   
 $du(8, result)$



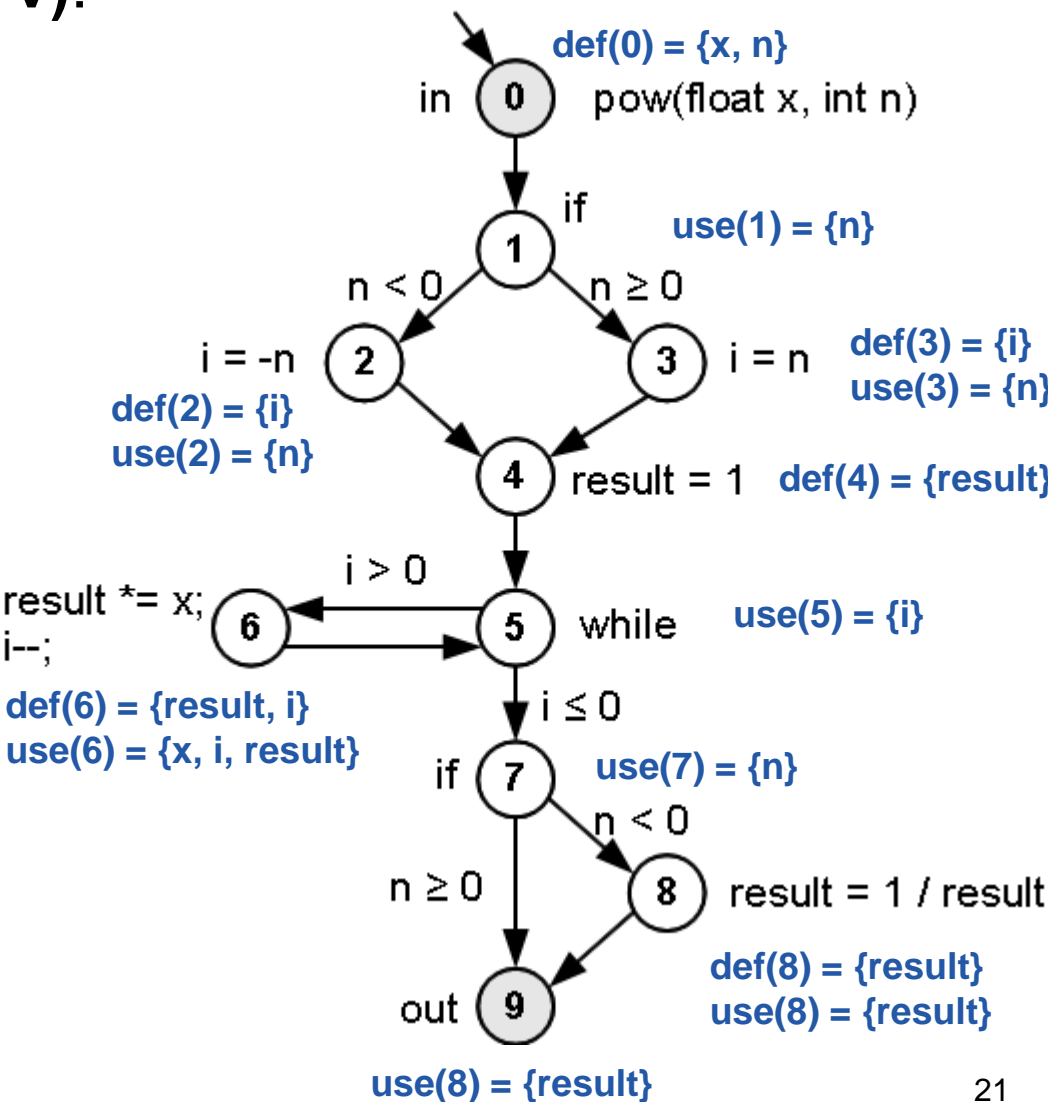


# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$   
=  $\{(0, 1, 2, 4, 5, 6), (0, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n)$   
=  $\{(0, 1)\}$   
 $du(2, 5, i)$   
=  $\{(2, 4, 5)\}$   
 $du(3, 5, i)$   
=  $\{(3, 4, 5)\}$   
 $du(4, 6, result)$   
=  $\{(4, 5, 6)\}$   
 $du(4, 8, result)$   
=  $\{(4, 5, 7, 8)\}$   
 $du(4, 9, result)$   
=  $\{(4, 5, 7, 9)\}$   
 $du(6, 5, i)$   
=  $\{(6, 5)\}$   
 $du(6, 6, result)$   
=  $\{(6, 5, 6)\}$   
 $du(6, 8, result)$   
=  $\{(6, 5, 7, 8)\}$   
 $du(6, 9, result)$   
=  $\{(6, 5, 7, 9)\}$   
 $du(8, result)$   
=  $\{(8, 9)\}$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

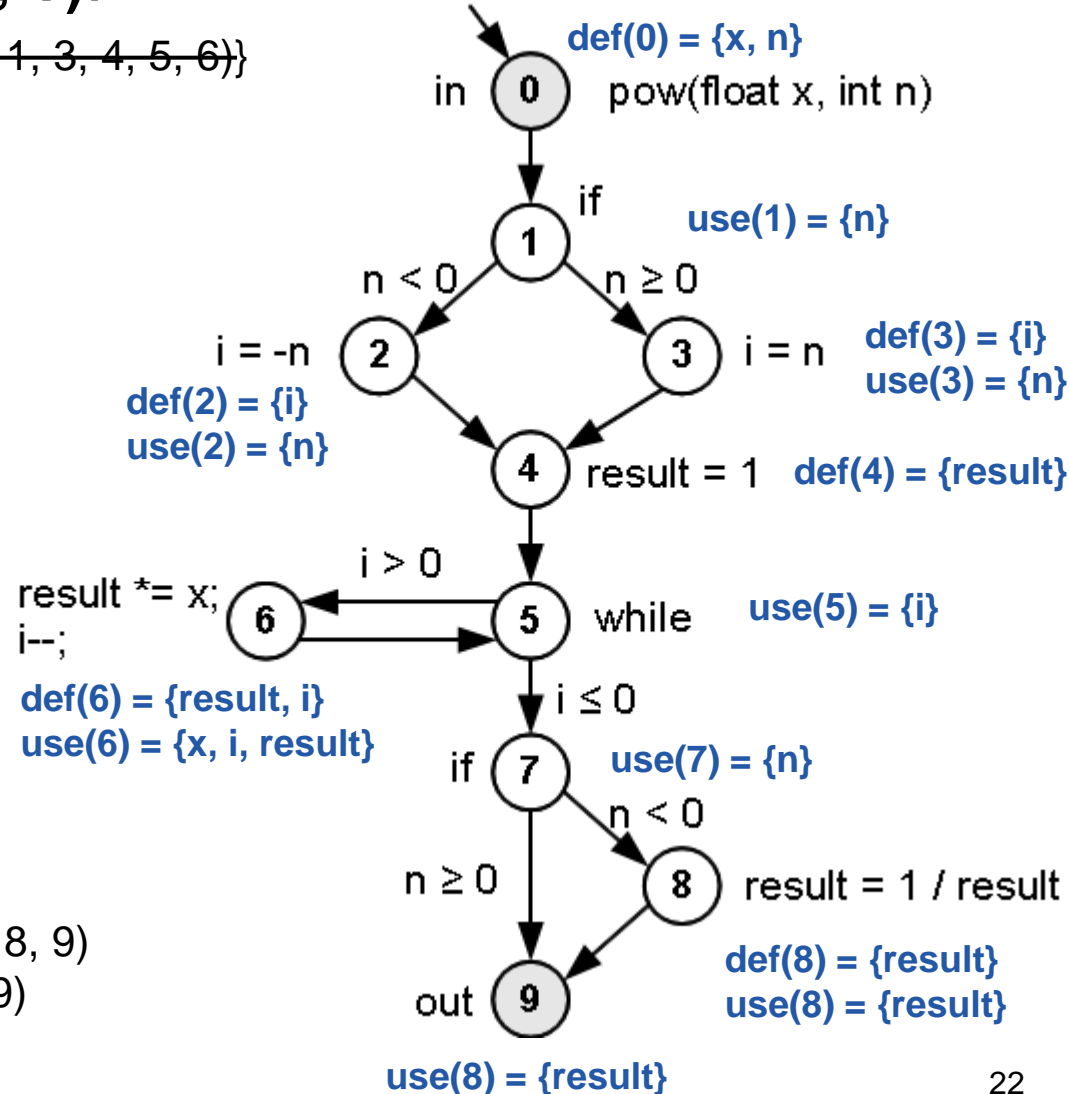
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$	$= \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)})\}$
$du(0, 1, n)$	$= \{(\underline{0}, \underline{1})\}$
$du(0, 7, n)$	$= \{(0, 1, 3, 4, 5, 7)\}$
$du(2, 5, i)$	$= \{(\underline{2}, \underline{4}, \underline{5})\}$
$du(3, 5, i)$	$= \{(\underline{3}, \underline{4}, \underline{5})\}$
$du(4, 6, result)$	$= \{(\underline{4}, \underline{5}, \underline{6})\}$
$du(4, 9, result)$	$= \{(4, 5, 7, 9)\}$
$du(6, 5, i)$	$= \{(\underline{6}, \underline{5})\}$
$du(6, 6, result)$	$= \{(6, 5, 6)\}$
$du(6, 8, result)$	$= \{(\underline{6}, \underline{5}, 7, 8)\}$
$du(6, 9, result)$	$= \{(\underline{6}, \underline{5}, 7, 9)\}$
$du(8, result)$	$= \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

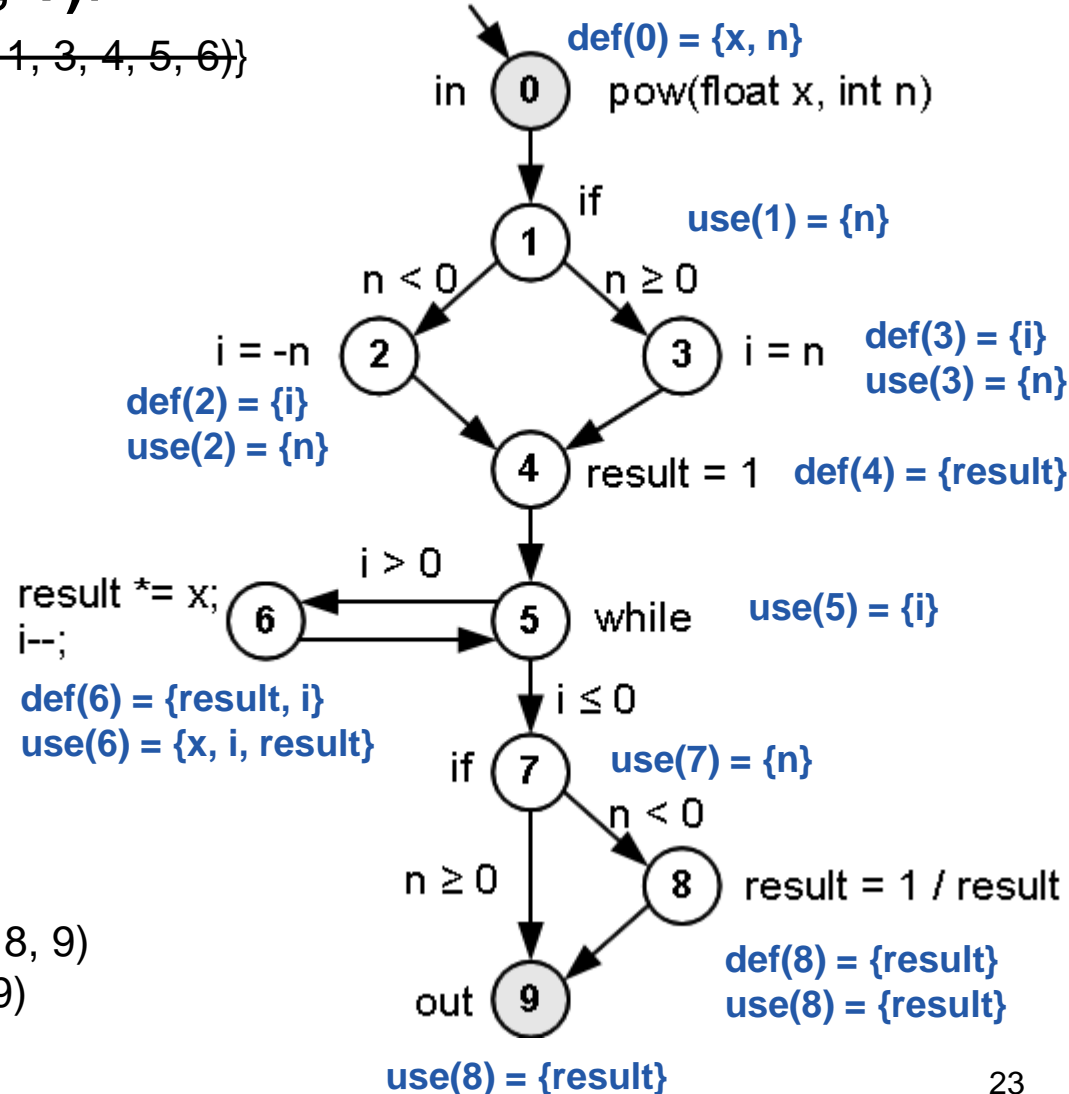
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$	$= \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)})\}$
$du(0, 1, n)$	$= \{(\underline{0}, \underline{1})\}$
$du(0, 7, n)$	$= \{(0, 1, 3, 4, 5, 7)\}$
$du(2, 5, i)$	$= \{(\underline{2}, \underline{4}, \underline{5})\}$
$du(3, 5, i)$	$= \{(\underline{3}, \underline{4}, \underline{5})\}$
$du(4, 6, result)$	$= \{(\underline{4}, \underline{5}, \underline{6})\}$
$du(4, 9, result)$	$= \{(4, 5, 7, 9)\}$
$du(6, 5, i)$	$= \{(\underline{6}, \underline{5})\}$
$du(6, 6, result)$	$= \{(6, 5, 6)\}$
$du(6, 8, result)$	$= \{(\underline{6}, \underline{5}, 7, 8)\}$
$du(6, 9, result)$	$= \{(\underline{6}, \underline{5}, 7, 9)\}$
$du(8, result)$	$= \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

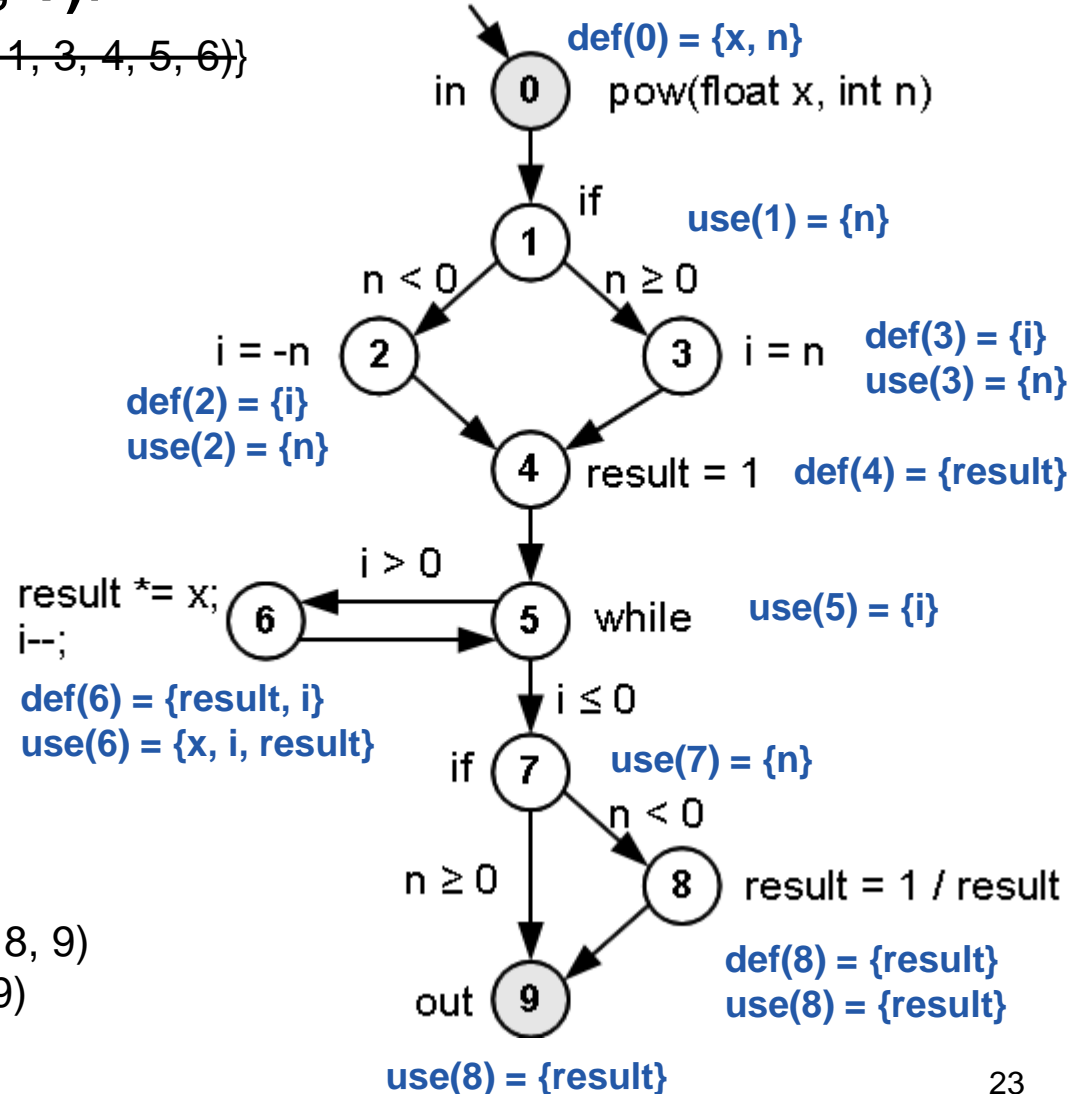
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$	$= \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)})\}$
$du(0, 1, n)$	$= \{(\underline{0}, \underline{1})\}$
$du(0, 7, n)$	$= \{(0, 1, 3, 4, 5, 7)\}$
$du(2, 5, i)$	$= \{(\underline{2}, \underline{4}, \underline{5})\}$
$du(3, 5, i)$	$= \{(\underline{3}, \underline{4}, \underline{5})\}$
$du(4, 6, result)$	$= \{(\underline{4}, \underline{5}, \underline{6})\}$
$du(4, 9, result)$	$= \{(\underline{4}, \underline{5}, \underline{7}, \underline{9})\}$
$du(6, 5, i)$	$= \{(\underline{6}, \underline{5})\}$
$du(6, 6, result)$	$= \{(6, 5, 6)\}$
$du(6, 8, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{8})\}$
$du(6, 9, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{9})\}$
$du(8, result)$	$= \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

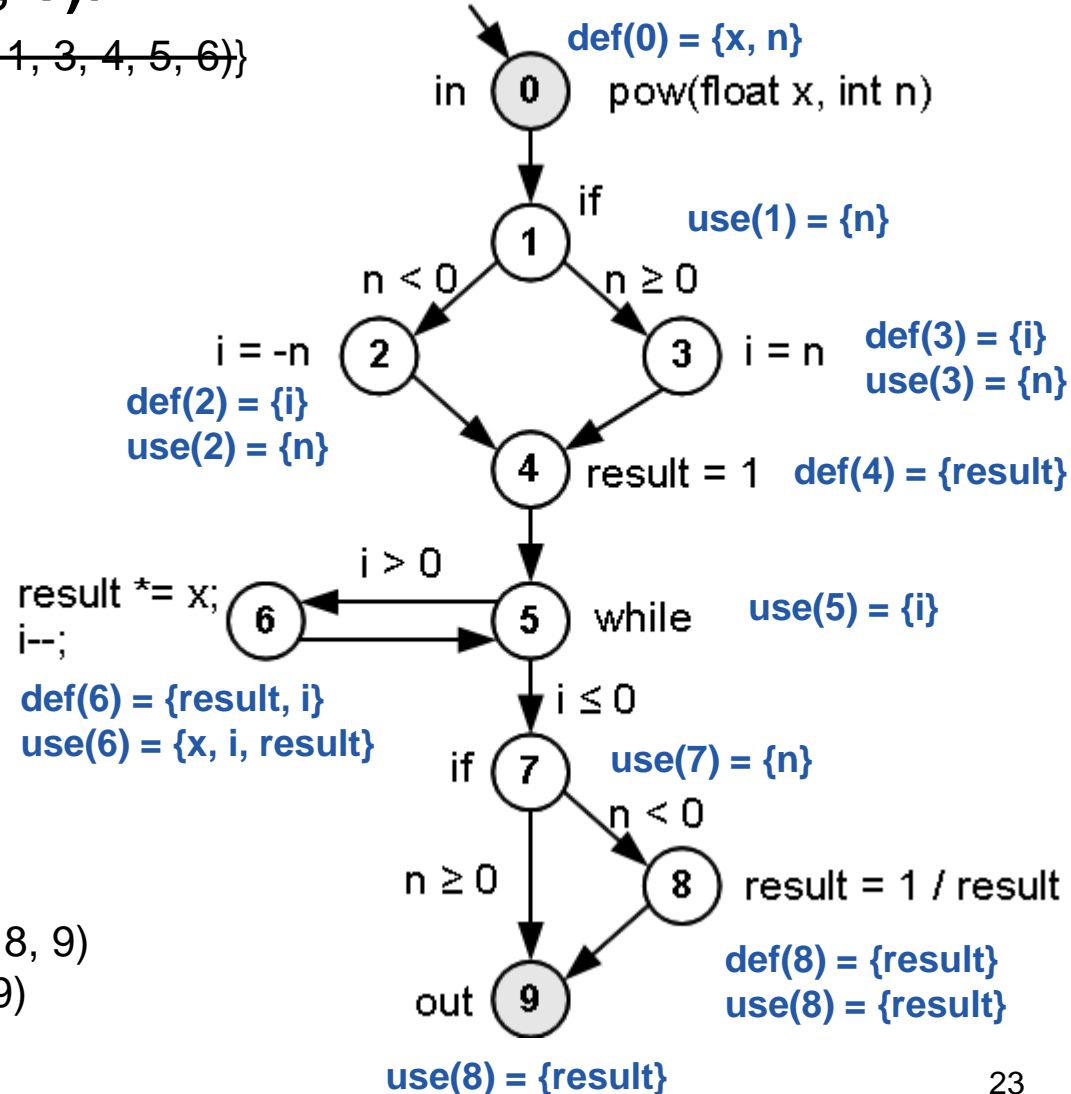
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x) = \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)}\}$   
 $du(0, 1, n) = \{(\underline{0}, \underline{1})\}$   
 $du(0, 7, n) = \{(\underline{0}, 1, 3, 4, 5, 7, \underline{)}\}$   
 $du(2, 5, i) = \{(\underline{2}, \underline{4}, \underline{5})\}$   
 $du(3, 5, i) = \{(\underline{3}, \underline{4}, \underline{5})\}$   
 $du(4, 6, result) = \{(\underline{4}, \underline{5}, \underline{6})\}$   
 $du(4, 9, result) = \{(\underline{4}, \underline{5}, 7, 9, \underline{)}\}$   
 $du(6, 5, i) = \{(\underline{6}, \underline{5})\}$   
 $du(6, 6, result) = \{(\underline{6}, 5, 6, \underline{)}\}$   
 $du(6, 8, result) = \{(\underline{6}, 5, 7, 8, \underline{)}\}$   
 $du(6, 9, result) = \{(\underline{6}, 5, 7, 9, \underline{)}\}$   
 $du(8, result) = \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$





# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

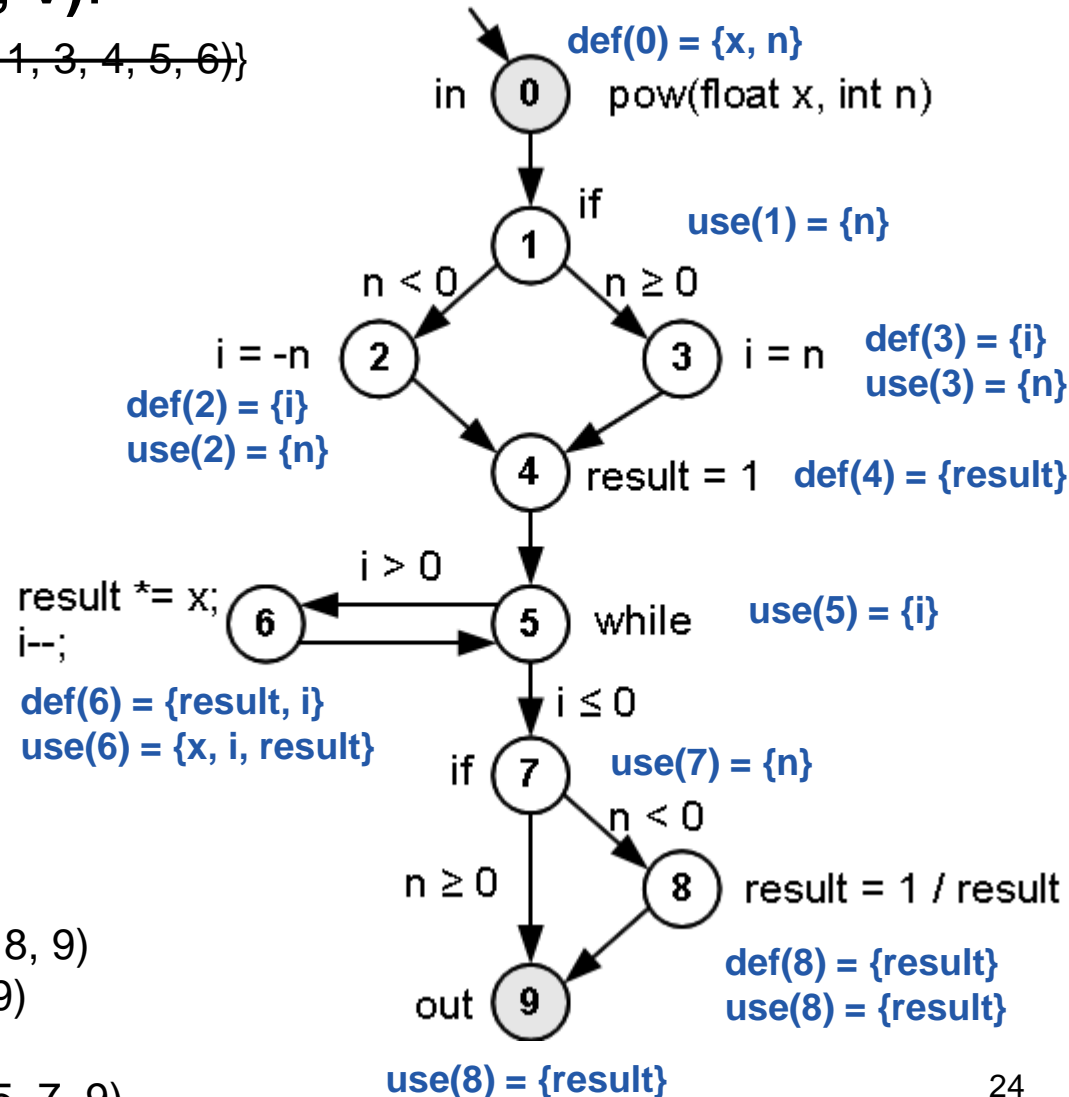
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$	$= \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)})\}$
$du(0, 1, n)$	$= \{(\underline{0}, \underline{1})\}$
$du(0, 7, n)$	$= \{(0, 1, 3, 4, 5, 7)\}$
$du(2, 5, i)$	$= \{(\underline{2}, \underline{4}, \underline{5})\}$
$du(3, 5, i)$	$= \{(\underline{3}, \underline{4}, \underline{5})\}$
$du(4, 6, result)$	$= \{(\underline{4}, \underline{5}, \underline{6})\}$
$du(4, 9, result)$	$= \{(\underline{4}, \underline{5}, \underline{7}, \underline{9})\}$
$du(6, 5, i)$	$= \{(\underline{6}, \underline{5})\}$
$du(6, 6, result)$	$= \{(6, 5, 6)\}$
$du(6, 8, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{8})\}$
$du(6, 9, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{9})\}$
$du(8, result)$	$= \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$   
 $pow(x, 2)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 6, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

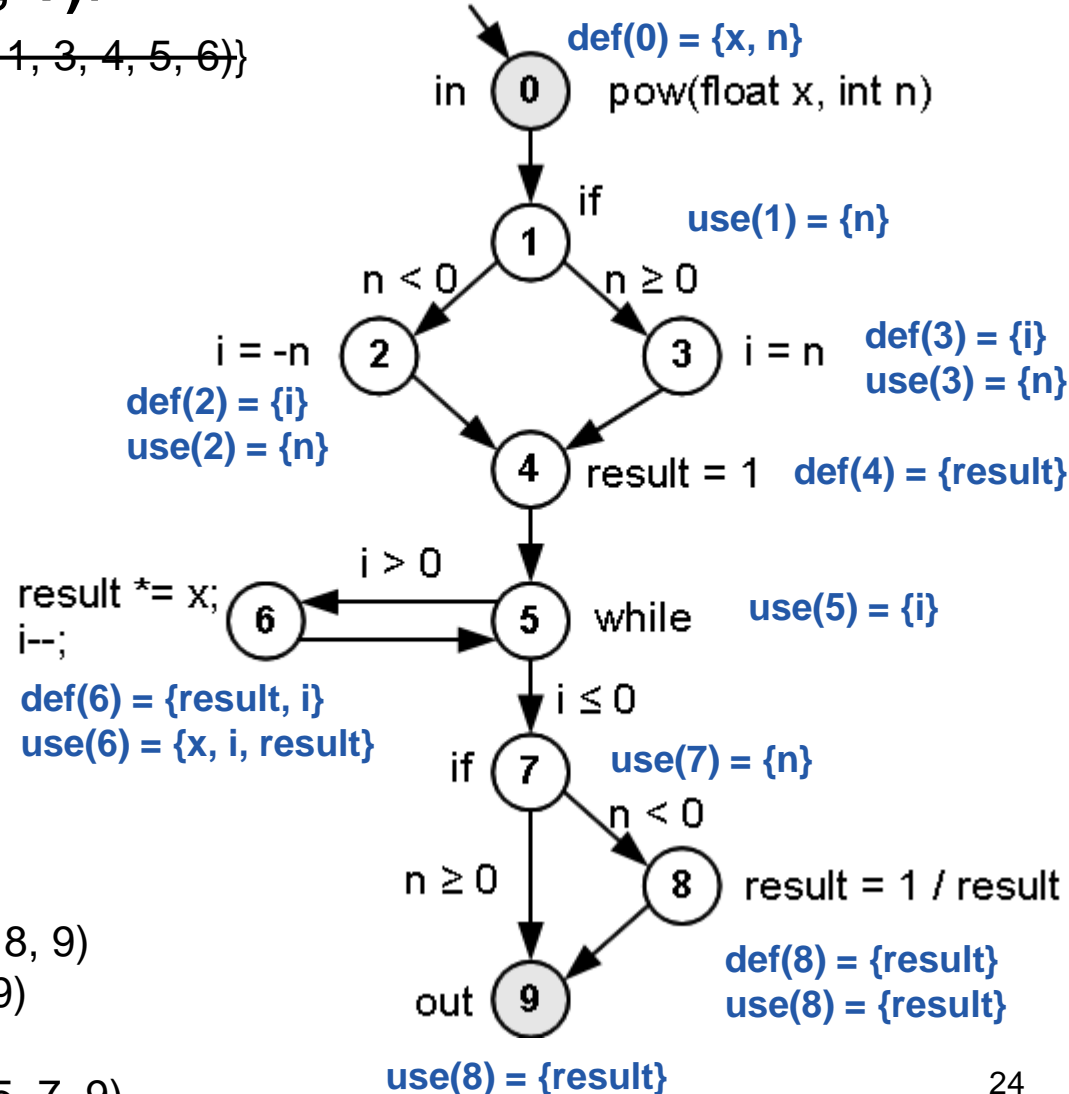
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x)$	$= \{(\underline{0}, 1, 2, 4, 5, 6, \underline{)}, (\underline{0}, 1, 3, 4, 5, 6, \underline{)})\}$
$du(0, 1, n)$	$= \{(\underline{0}, \underline{1})\}$
$du(0, 7, n)$	$= \{(0, 1, 3, 4, 5, 7)\}$
$du(2, 5, i)$	$= \{(\underline{2}, \underline{4}, \underline{5})\}$
$du(3, 5, i)$	$= \{(\underline{3}, \underline{4}, \underline{5})\}$
$du(4, 6, result)$	$= \{(\underline{4}, \underline{5}, \underline{6})\}$
$du(4, 9, result)$	$= \{(\underline{4}, \underline{5}, \underline{7}, \underline{9})\}$
$du(6, 5, i)$	$= \{(\underline{6}, \underline{5})\}$
$du(6, 6, result)$	$= \{(\underline{6}, \underline{5}, \underline{6})\}$
$du(6, 8, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{8})\}$
$du(6, 9, result)$	$= \{(\underline{6}, \underline{5}, \underline{7}, \underline{9})\}$
$du(8, result)$	$= \{(\underline{8}, \underline{9})\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$   
 $pow(x, 2)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 6, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-Uses – Beispiel

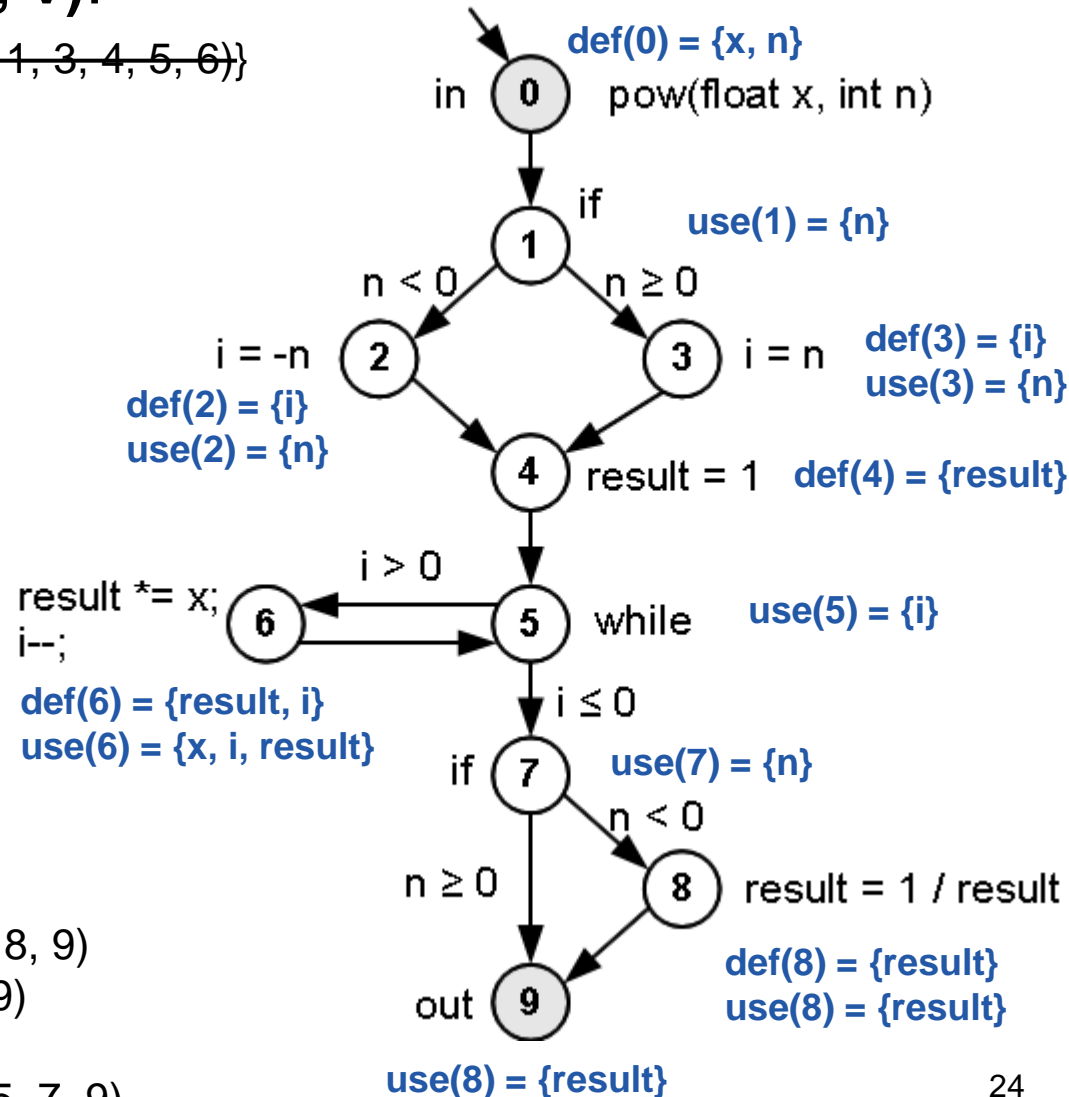
- Alle Pfadmenge  $du(n, n', v)$ :

$du(0, 6, x) = \{(\underline{0}, 1, 2, 4, 5, 6), (\underline{0}, 1, 3, 4, 5, 6)\}$   
 $du(0, 1, n) = \{(\underline{0}, 1)\}$   
 $du(0, 7, n) = \{(\underline{0}, 1, 3, 4, 5, 7)\}$   
 $du(2, 5, i) = \{(\underline{2}, 4, 5)\}$   
 $du(3, 5, i) = \{(\underline{3}, 4, 5)\}$   
 $du(4, 6, result) = \{(\underline{4}, 5, 6)\}$   
 $du(4, 9, result) = \{(\underline{4}, 5, 7, 9)\}$   
 $du(6, 5, i) = \{(\underline{6}, 5)\}$   
 $du(6, 6, result) = \{(\underline{6}, 5, 6)\}$   
 $du(6, 8, result) = \{(\underline{6}, 5, 7, 8)\}$   
 $du(6, 9, result) = \{(\underline{6}, 5, 7, 9)\}$   
 $du(8, result) = \{(\underline{8}, 9)\}$

- All-Uses**

– Überdeckung durch folgende Tests:

$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$   
 $pow(x, 2)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 6, 5, 7, 9)$





# Defs-Uses-Überdeckungskriterien

## Varianten von All-Uses

---

- Varianten von **All-Uses**:
  - **Für alle Definitionen einer Variable**, überdecke mindestens einen bezüglich dieser Variablen definitionsfreien Pfad...
- **All-c-Uses**:
  - ... zu **jedem** c-use
- **All-p-Uses**:
  - ... zu **jedem** p-use
- **All-c-some-p**:
  - ... zu **jedem** c-use. Gibt es keinen def.freien Pfad zu einem c-use, überdecke mindestens def.freien Pfad zu **einem** p-use
- **All-p-some-c**:
  - ... zu **jedem** p-use. Gibt es keinen def.freien Pfad zu einem p-use, überdecke mindestens def.freien Pfad zu **einem** c-use



# Defs-Uses-Überdeckungskriterien

## All-DU-Paths

---

- Überdeckungskriterium: **All-DU-Paths**
  - Überdecke alle DU-Pfade
  - Etwas formaler: **Für jede Pfadmenge  $du(n, v)$  müssen alle Pfade aus dieser Menge überdeckt werden**



# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Alle Pfadmenge  $du(n, n', v)$ :

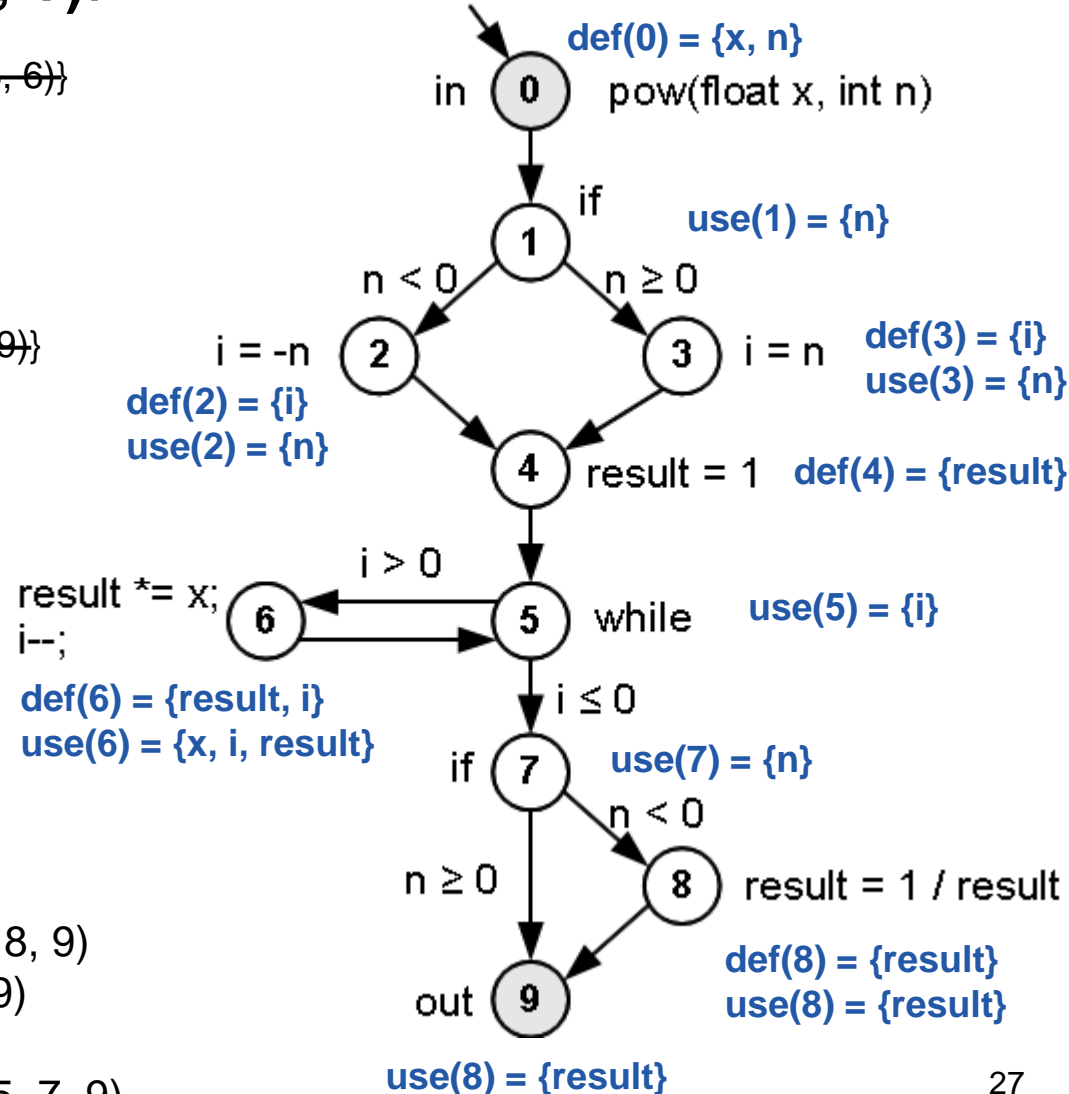
$du(0, x) = \{(\overline{0, 1, 2, 4, 5, 6,}), (\overline{0, 1, 3, 4, 5, 6})\}$   
 $du(0, n) = \{(\overline{0, 1})\}$   
 $du(2, i) = \{(\overline{2, 4, 5})\}$   
 $du(3, i) = \{(\overline{3, 4, 5})\}$   
 $du(4, result) = \{(\overline{4, 5, 6}), (\overline{4, 5, 7, 9})\}$   
 $du(6, i) = \{(\overline{6, 5})\}$   
 $du(6, result) = \{(\overline{6, 5, 6}), (\overline{6, 5, 7, 8}), (\overline{6, 5, 7, 9})\}$   
 $du(8, result) = \{(\overline{8, 9})\}$

In diesem Fall schon  
alles erledigt

- All-DU-Paths

– Überdeckung durch  
folgende Tests:

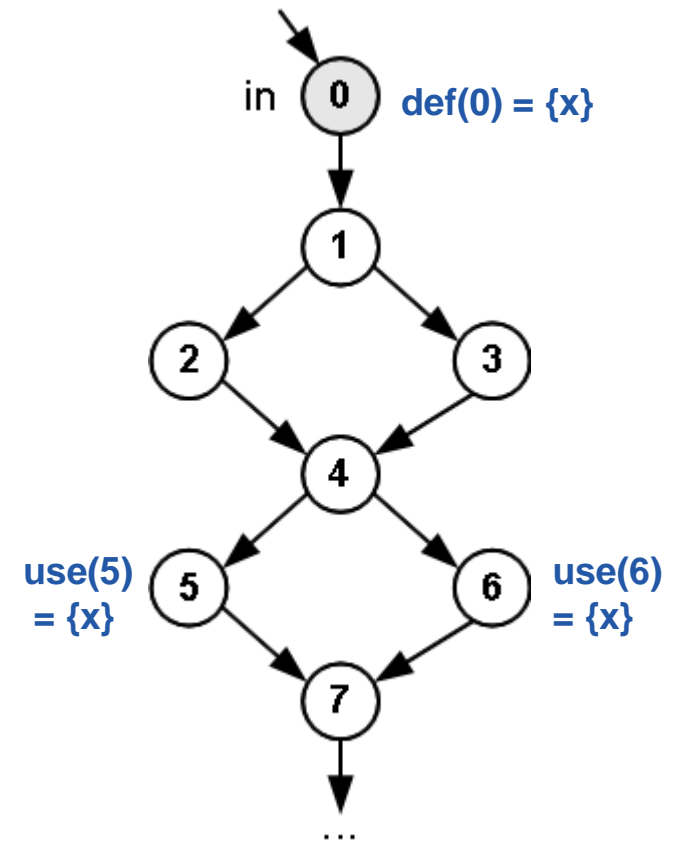
$pow(x, -1)$  überdeckt  $(0, 1, 2, 4, 5, 6, 5, 7, 8, 9)$   
 $pow(x, 1)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 7, 9)$   
 $pow(x, 0)$  überdeckt  $(0, 1, 3, 4, 5, 7, 9)$   
 $pow(x, 2)$  überdeckt  $(0, 1, 3, 4, 5, 6, 5, 6, 5, 7, 9)$



# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...

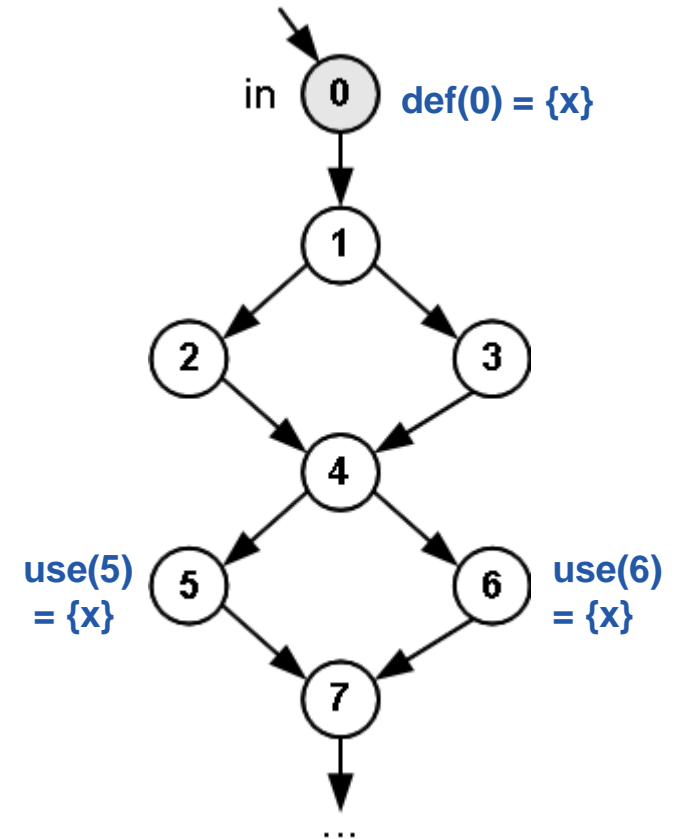




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**



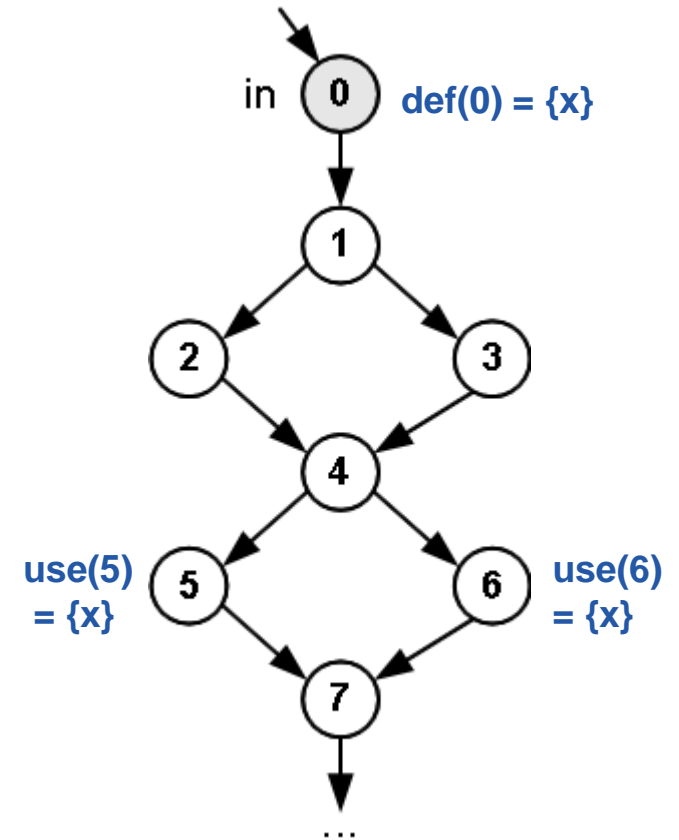




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:

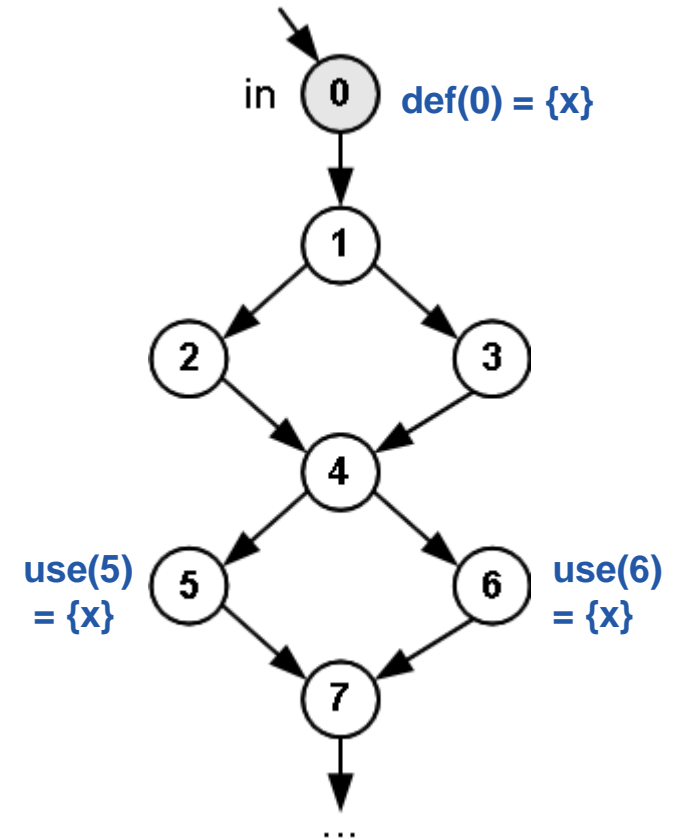




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)

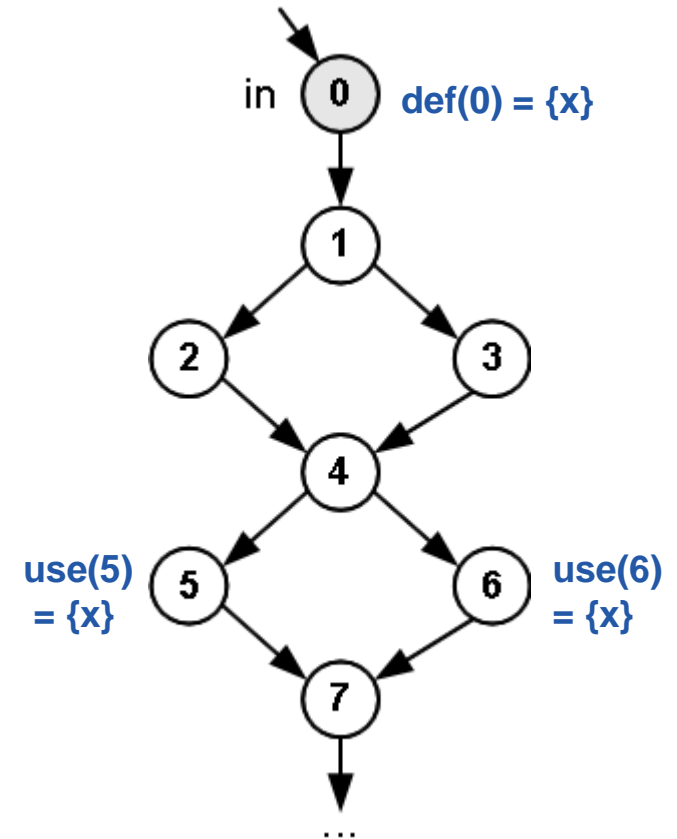




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**

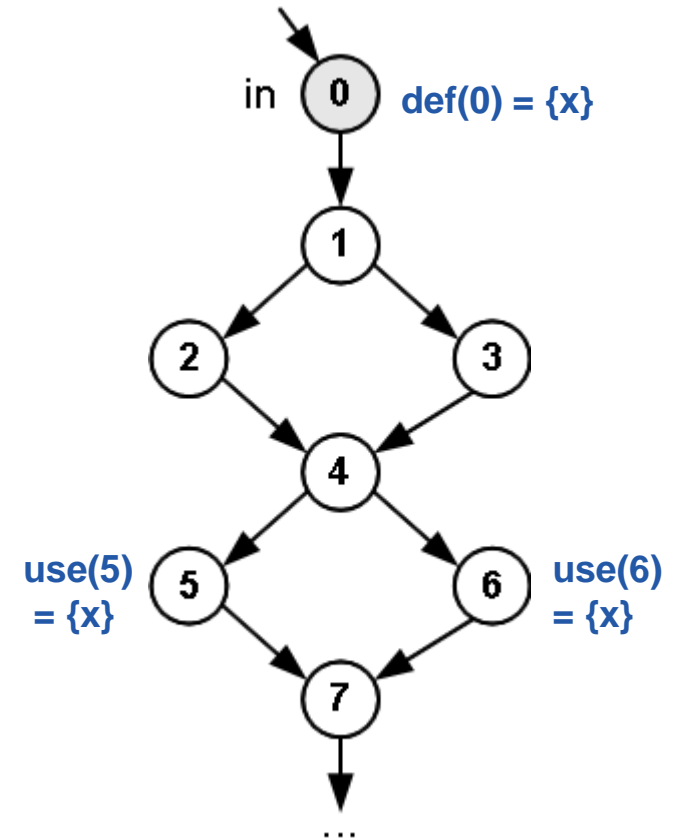




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**
  - Überdeckung von **einem** Pfad von 0 zu 5 **und** **einem** Pfad von 0 zu 6:

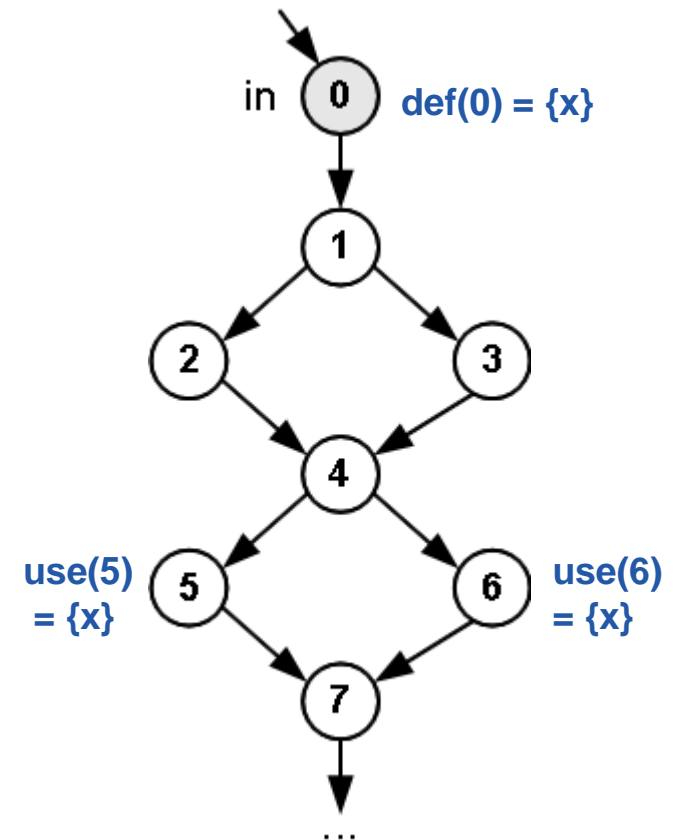




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**
  - Überdeckung von **einem** Pfad von 0 zu 5 **und** **einem** Pfad von 0 zu 6:
    - Z.B. (0, 1, 2, 4, 5), (0, 1, 2, 4, 6)

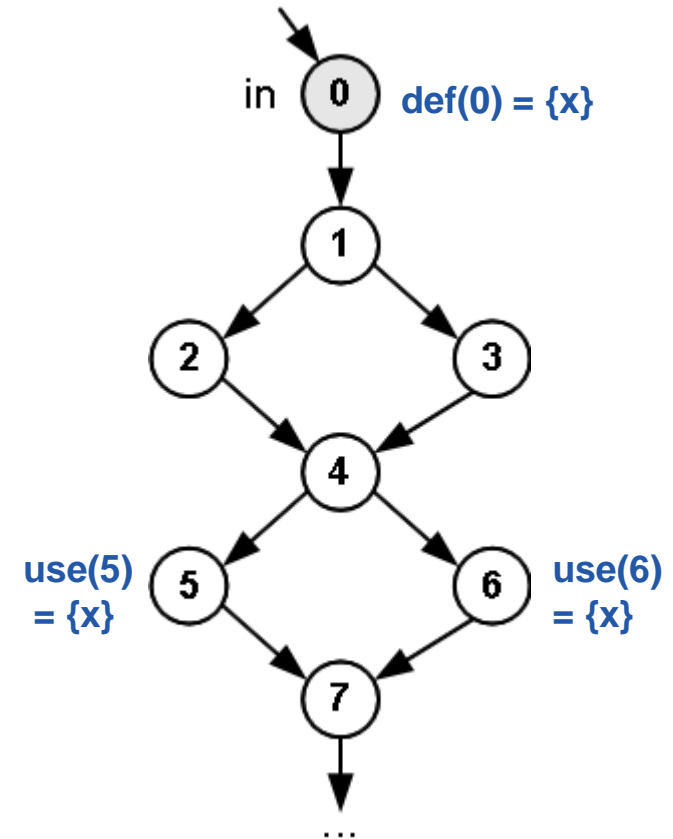




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**
  - Überdeckung von **einem** Pfad von 0 zu 5 **und** **einem** Pfad von 0 zu 6:
    - Z.B. (0, 1, 2, 4, 5), (0, 1, 2, 4, 6)
- **All-DU-Paths**

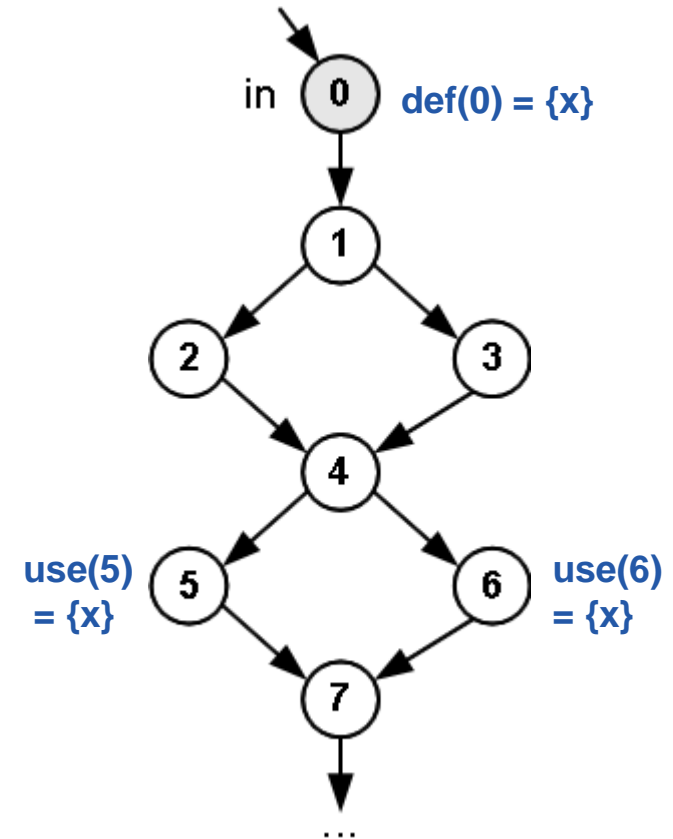




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**
  - Überdeckung von **einem** Pfad von 0 zu 5 **und** **einem** Pfad von 0 zu 6:
    - Z.B. (0, 1, 2, 4, 5), (0, 1, 2, 4, 6)
- **All-DU-Paths**
  - Überdeckung von **jedem** Pfad von 0 zu 5 **und** **jedem** Pfad von 0 zu 6:

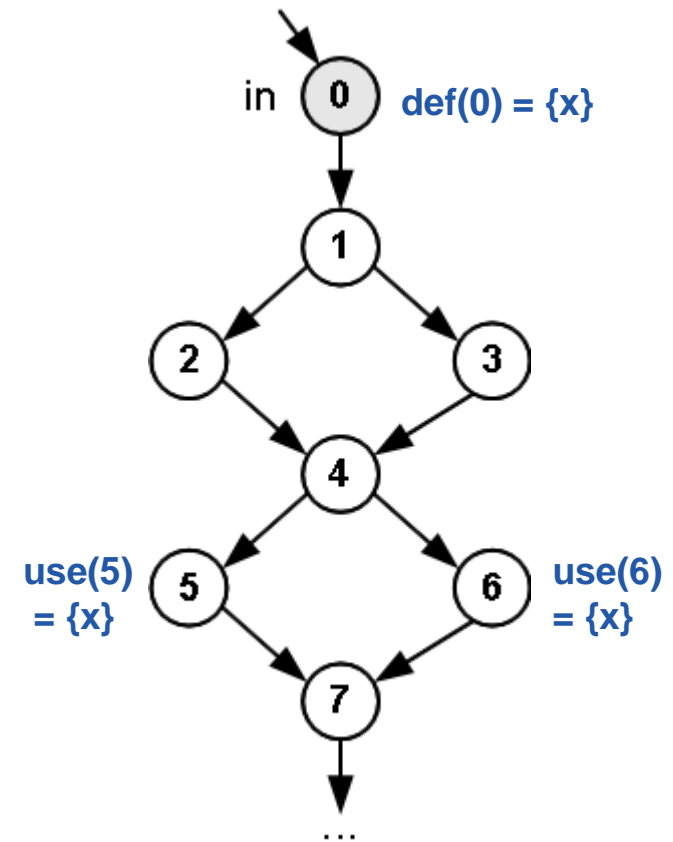




# Defs-Uses-Überdeckungskriterien

## All-DU-Paths – Beispiel

- Zu überdeckende Pfade für...
- **All-Defs**
  - Überdeckung von **einem** Pfad von 0 zu 5 **oder** 6:
    - Z.B. (0, 1, 2, 4, 5)
- **All-Uses**
  - Überdeckung von **einem** Pfad von 0 zu 5 **und** **einem** Pfad von 0 zu 6:
    - Z.B. (0, 1, 2, 4, 5), (0, 1, 2, 4, 6)
- **All-DU-Paths**
  - Überdeckung von **jedem** Pfad von 0 zu 5 **und** **jedem** Pfad von 0 zu 6:
    - (0, 1, 2, 4, 5), (0, 1, 3, 4, 5)
    - (0, 1, 2, 4, 6), (0, 1, 3, 4, 6)

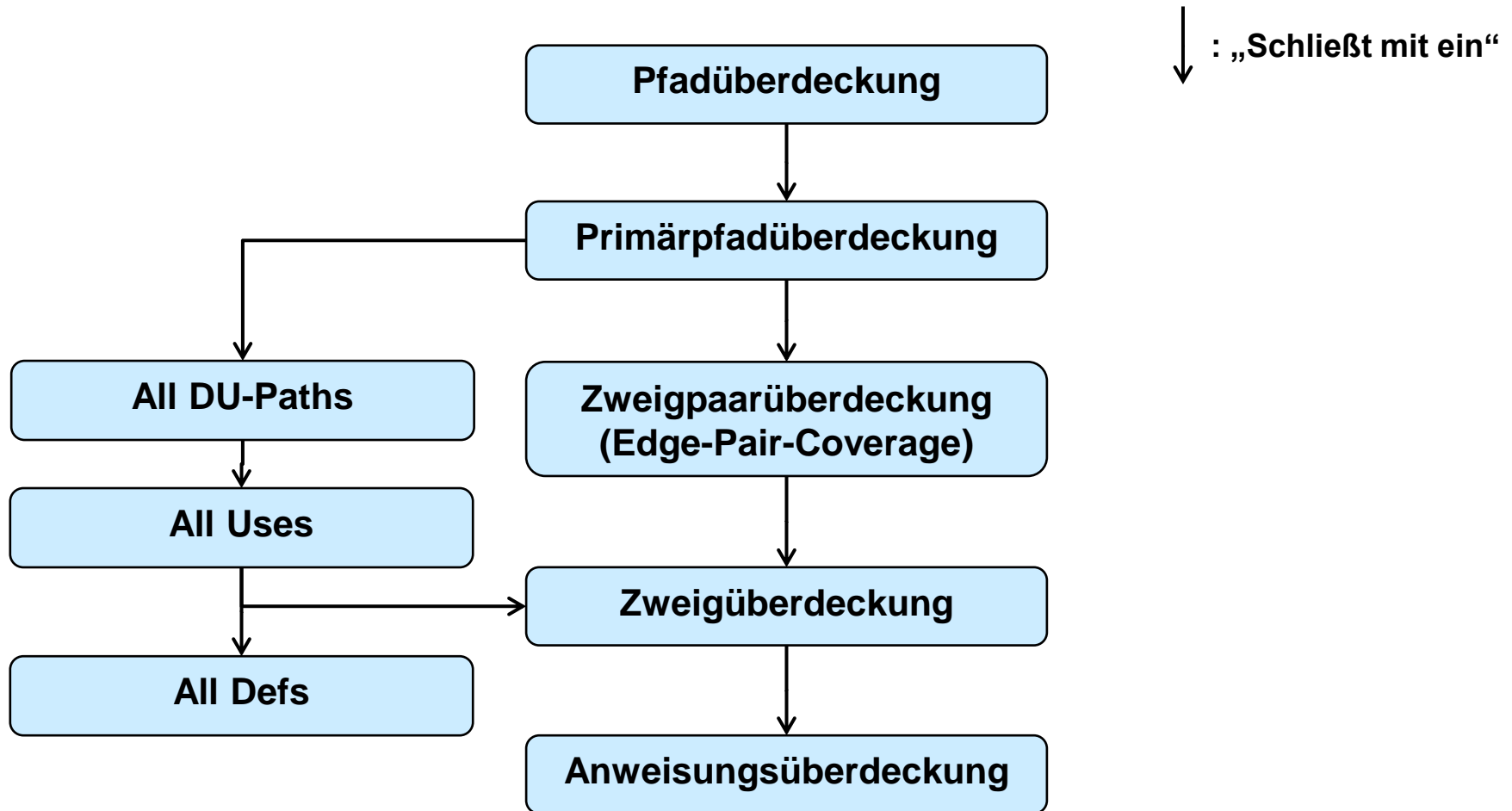






# Datenflussorientierte Überdeckungskriterien im Vergleich

- Die Überdeckungskriterien subsumieren sich wie folgt:





# Mutationstests

---

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich



# Mutationstests

---

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe
- **Idee 2:** Erzeuge **Mutanten** des Programms: Versionen mit (zufällig oder systematisch) eingefügten Defekten

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe
- **Idee 2:** Erzeuge **Mutanten** des Programms: Versionen mit (zufällig oder systematisch) eingefügten Defekten
  - Identifizieren die Tests alle Mutanten, dann scheinen meine Tests effektiv zu sein

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe
- **Idee 2:** Erzeuge **Mutanten** des Programms: Versionen mit (zufällig oder systematisch) eingefügten Defekten
  - Identifizieren die Tests alle Mutanten, dann scheinen meine Tests effektiv zu sein
    - Erstmal Vermutung – aber in der Praxis bewahrheitet!

- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe
- **Idee 2:** Erzeuge **Mutanten** des Programms: Versionen mit (zufällig oder systematisch) eingefügten Defekten
  - Identifizieren die Tests alle Mutanten, dann scheinen meine Tests effektiv zu sein
    - Erstmal Vermutung – aber in der Praxis bewahrheitet!
  - Falls Mutanten unentdeckt geblieben sind, erstelle ich neue Testfälle, die diese Mutanten erfolgreich identifizieren („töten“)



- Gegeben sei eine Programm und eine Testmenge
  - Angenommen, alle Tests sind erfolgreich
- **Frage:** Wie sicher kann ich mir sein, dass keine Fehler mehr im Programm enthalten sind?
  - **Idee 1:** Ich bestimme zu welchem Grad ich ein bestimmtes Überdeckungskriterium erfüllt habe
- **Idee 2:** Erzeuge **Mutanten** des Programms: Versionen mit (zufällig oder systematisch) eingefügten Defekten
  - Identifizieren die Tests alle Mutanten, dann scheinen meine Tests effektiv zu sein
    - Erstmal Vermutung – aber in der Praxis bewahrheitet!
  - Falls Mutanten unentdeckt geblieben sind, erstelle ich neue Testfälle, die diese Mutanten erfolgreich identifizieren („töten“)
  - Somit steigere ich die Effektivität der Testmenge



- Es gibt viele verschiedene Mutationsoperatoren, z.B.

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

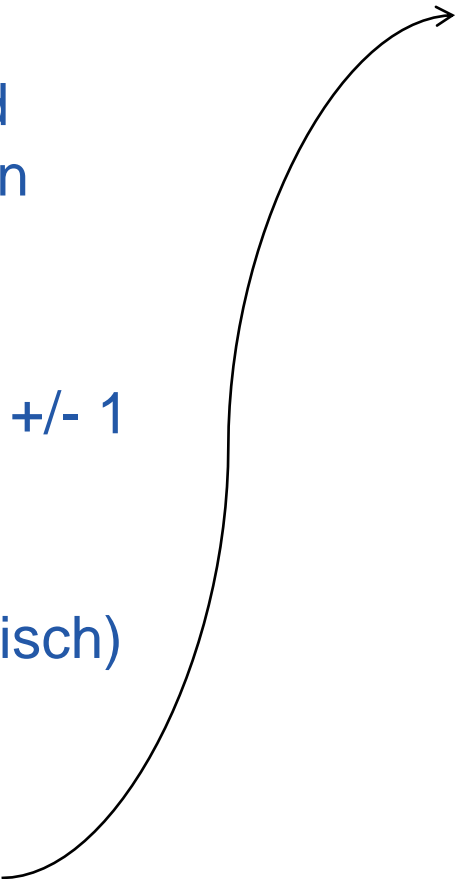
- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen
  - Viele mehr... (Manche programmiersprachenspezifisch)

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen
  - Viele mehr... (Manche programmiersprachenspezifisch)

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a > 0 || b > 0) {  
    x = x + 10;  
}  
...
```



- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen
  - Viele mehr... (Manche programmiersprachenspezifisch)

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a > 0 || b > 0) {  
    x = x + 10;  
}  
...  
...  
if (a < 0 && b > 0) {  
    x = x + 10;  
}  
...
```



- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen
  - Viele mehr... (Manche programmiersprachenspezifisch)

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a > 0 || b > 0) {  
    x = x + 10;  
}  
...  
...  
if (a < 0 && b > 0) {  
    x = x + 10;  
}  
...  
...  
if (a > 0 && b > 0) {  
    x = x - 10;  
}  
...
```

- Es gibt viele verschiedene Mutationsoperatoren, z.B.
  - Logische, Arithmetische und Vergleichsoperatoren ändern
  - Variablen vertauschen oder durch Konstanten ersetzen
  - Konstanten durch 0, 1, -1, c +/- 1 ersetzen
  - Viele mehr... (Manche programmiersprachenspezifisch)

```
...  
if (a > 0 && b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a > 0 || b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a < 0 && b > 0) {  
    x = x + 10;  
}  
...
```

```
...  
if (a > 0 && b > 0) {  
    x = x - 10;  
}  
...
```

```
...  
if (a > 0 && b > 0)  
{  
    x = x + 11;  
}  
...
```



# Mutationstests

---

- Manchmal ändern Mutationen nichts am Verhalten des Programms
  - Funktional gleich
  - Ggf. veränderte Laufzeit

- Manchmal ändern Mutationen nichts am Verhalten des Programms
  - Funktional gleich
  - Ggf. veränderte Laufzeit
- Beispiel: Funktional gleich:

- Manchmal ändern Mutationen nichts am Verhalten des Programms
  - Funktional gleich
  - Ggf. veränderte Laufzeit
- Beispiel: Funktional gleich:

Program $p$	Equivalent Mutant $m$
<pre>for (<i>int</i> <math>i = 0</math>; <math>i &lt; 10</math>; <math>i++</math>) {   ...(<i>the value of i</i>     <i>is not changed</i>) }</pre>	<pre>for (<i>int</i> <math>i = 0</math>; <math>i \neq 10</math>; <math>i++</math>) {   ...(<i>the value of i</i>     <i>is not changed</i>) }</pre>