10. Kapitel

Optimierung
Teil 1

Datenflußanalyse

Kapitel 10: Datenflußanalyse

0. Einbettung
1. DFA-Schemata
2. Theorie
   - CPO, Fixpunktsatz
   - Monotone Datenflußrahmen
   - MFP & MPO
3. Konkrete Analysen
   - 3.1 Erreichende Definitionen
   - 3.2 Gültige Ausdrücke
4. Abstraktionen der Ablaufsteuerung
   - 4.1 Intervallanalyse
   - 4.2 Dominanzanalyse
   - 4.3 Eigenschaften der DG

10. Datenflußanalyse (DFA)

Teil der Verfahren zur Programmanalyse, z.B.

Nielson, Nielson, Hankin: *Principles of Program Analysis*. Springer 1999.

Aufgabe: Ermittlung von Zusammenhängen (Datenabhängigkeiten) zwischen Programmstellen (Tripeln) und den dort berechneten Werten unter Berücksichtigung der möglichen Programmabläufe

- damit indirekt Beschränkung möglicher Befehlsanordnungen:
  - Berechnung t kommt initial vor t', $t < t'$, wenn t vor t' nach der Semantik des Quellprogramms kommt
  - t kommt wesentlich vor t', $t << t'$, wenn t' datenabhängig von t ist, d.h. Vertauschung könnte Ergebnisse verändern

Einsatz in

- Übersetzerbau: Grundlage der globalen Optimierung
- Programmanalyse im *software engineering*: Programmeigenschaften bestimmen, z.B. nicht initialisierte Variable; Jahr2000-Problem; Reengineering; ...

10. Beispielaufgaben

Wenn t ein Kalenderdatum berechnet, welche anderen Werte sind auch ein Datum
(allgemeiner: haben gleichen „Typ" wie t)?

Ausführungspfad gegeben: Welche Zuweisungen schreiben
in welcher Reihenfolge in eine gegebene Speicherzelle?

Welche anderen Variablen tragen zum Wert von v
an einer bestimmten Programmstelle bei?

definiert-benutzt-Beziehung (*def-use*): Wenn Tripel t an v zuweist:
Wo wird der Wert benutzt?

benutzt-definiert-Beziehung (*use-def*): Wenn Tripel t v als Operand benutzt:
welche Zuweisungen an v könnten dessen Wert definieren?

Lebendigkeit an gegebener Programmstelle:
Welche schon definierten Werte werden (möglicherweise) noch benutzt?

...

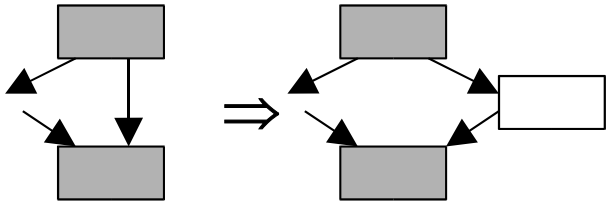
10. Programmrepräsentation für DFA

allgemein: Ablaufgraph, z.B. Flußdiagramm, Tripel- oder SSA-Darstellung

- iterative Lösung der Datenflußgleichungen
- Anzahl Iterationen von Reihenfolge der Bearbeitung der Grundblöcke und Struktur des Ablaufgraphen abhängig

bei `goto`-freien Sprachen: DFA effizient durch Attributierung des Strukturbaums berechenbar

- aber: für den Übersetzerbau unbrauchbar, da Adreßrechnung noch nicht explizit



Annahmen:

- kritische Kanten: Einfügen leere Ecke
- Programm (Prozedur) beginnt mit einer(!) Startecke S, endigt mit einer(!) Endecke E
- jede Anweisung von S erreichbar
- E von jeder Anweisung erreichbar

Kapitel 10: Datenflußanalyse

0. Einbettung
1. DFA-Schemata
2. Theorie
   - CPO, Fixpunktsatz
   - Monotone Datenflußrahmen
   - MFP & MPO
3. Konkrete Analysen
   - 3.1 Erreichende Definitionen
   - 3.2 Gültige Ausdrücke
4. Abstraktionen der Ablaufsteuerung
   - 4.1 Intervallanalyse
   - 4.2 Dominanzanalyse
   - 4.3 Eigenschaften der DG

10.1 Datenflußgleichungen: Vorgaben

- gegeben . . . $M : A; N : \ldots$
- M, N Programmstellen
- A Anweisung, Tripel oder Grundblock
- $E(M)$, $E(N)$ Eigenschaften an Programmstellen M, N
- $E(\ldots)$ aufgefaßt als Menge oder boolesche Aussage
- Dualitäten: vorwärts - rückwärts, sicher (*must*) - möglich (*may*)

10.1 Vier DFA-Schemata

- vorwärts und sicher:

$$E_{in}(A) = \bigcap_{X \in Vor(A)} E_{out}(X)$$
$$E_{out}(A) = E_{in}(A) - kill(A) \cup gen(A)$$

- rückwärts und sicher:

$$E_{out}(A) = \bigcap_{X \in Nach(A)} E_{in}(X)$$
$$E_{in}(A) = E_{out}(A) - kill(A) \cup gen(A)$$

- vorwärts und möglich:

$$E_{in}(A) = \bigcup_{X \in Vor(A)} E_{out}(X)$$
$$E_{out}(A) = E_{in}(A) - kill(A) \cup gen(A)$$

- rückwärts und möglich:

$$E_{out}(A) = \bigcup_{X \in Nach(A)} E_{in}(X)$$
$$E_{in}(A) = E_{out}(A) - kill(A) \cup gen(A)$$

$kill(A)$: Eigenschaften E, durch A beseitigt

$gen(A)$: Eigenschaften E, durch A neu definiert (aufgabenspezifisch)

10.1 Initialisierung und Ergebnis

sicher:

- Initialisierung mit U
- Ergebnis größter Fixpunkt

möglich:

- Initialisierung mit $\emptyset$
- Ergebnis kleinster Fixpunkt

vorwärts nicht definiert für Startknoten

rückwärts nicht definiert für Endknoten

10.1 Beispiele

Erreichende Definitionen (reaching definitions):
Welche „Definitionen" (Zuweisungen) sind bei N gültig? (vorwärts, sicher)

Lebendige Variable (live variables): Welche Variablen werden vor einer Neuzuweisung noch benutzt? (rückwärts, möglich)

Verfügbare Ausdrücke (available expressions):
Welche (Zwischen-) Ergebnisse sind noch gültig? (vorwärts, sicher)

Teilweise verfügbare Ausdrücke (partially available expressions):
Welche (Zwischen-) Ergebnisse sind auf manchen Wegen berechnet worden und noch gültig? (vorwärts, möglich)

Wichtige Ausdrücke (busy expressions):
Welche Ausdrücke werden auf jeden Fall noch benutzt? (rückwärts, sicher)

Konstantenweitergabe (constant propagation): Welche Variablen haben einen bekannten, konstanten Wert bei N? (vorwärts, sicher)

Kapitel 10: Datenflußanalyse

0. Einbettung
1. DFA-Schemata
2. Theorie
   - CPO, Fixpunktsatz
   - Monotone Datenflußrahmen
   - MFP & MPO
3. Konkrete Analysen
   - 3.1 Erreichende Definitionen
   - 3.2 Gültige Ausdrücke
4. Abstraktionen der Ablaufsteuerung
   - 4.1 Intervallanalyse
   - 4.2 Dominanzanalyse
   - 4.3 Eigenschaften der DG

10.2 Vollständige Halbordnung
Complete Partial Order (CPO)

- Halbgeordnete Menge $(U, \leq)$
- Kleinstes Element $\perp$
- Jede aufsteigende Kette $K \subseteq U$ hat obere Grenze $\sup(K)$

Funktion f: $U \rightarrow U$ heißt

monoton: $x \leq y \Rightarrow f(x) \leq f(y)$ mit $x, y \in U$

stetig: $f(\sup(K)) = \sup(f(K))$ mit

$$f(K) = f([x_1, x_2, \ldots]) = [f(x_1), f(x_2), \ldots]$$

Es gilt: f stetig $\Rightarrow$ f monoton, f monoton $\wedge$ U endlich $\Rightarrow$ f stetig

10.2 Fixpunktsatz (Knaster-Tarski)

Für $CPO\ (U, \leq)$ und monotone Funktion f: $U \to U$ gilt

kleinster Fixpunkt X mit $f\ (X) = X$ existiert

X ist eindeutig

Fixpunktsatz für stetige f:

Für $CPO\ (U, \leq)$ mit kleinstem Element $\perp$ und stetige Funktion f: $U \to U$ gilt

kleinster Fixpunkt $X = \sup\{f^n(\perp)\}$

X iterativ berechenbar

dual:

Fixpunktsatz für stetige, monoton fallende f und größtem Element U:

Für $CPO\ (U, \geq)$ mit größtem Element U und stetige fallende Funktion f: $U \to U$:

größter Fixpunkt $X = \inf\{f^n(U)\}$

X iterativ berechenbar

10.2 Monotone Datenflußrahmen

- Lösung von Datenflußgleichungen ist Fixpunktberechnung
- Existenz des kleinsten Fixpunkts X ist gesichert, wenn Wertebereich U der Eigenschaften $E(A)$ vollständig halbgeordnet $(U, \leq)$
- Überschätzung Y des Fixpunkts X, $Y>X$, führt zu **ungenaueren** Schlüssen: nicht alle verfügbare Information berücksichtigt
- Unterschätzung Y des Fixpunkts X, $Y<X$, führt zu **Fehlern**: schärfere Annahme als zulässig
- Wertebereich U ist oft ein Verband, Spezialfälle:
  - Boolescher Verband $\{0,1\}$
  - Bitvektorverband $\{0,1\}^n$
- Wirkung der Anweisungen A modelliert durch monotone Transferfunktionen $t: U \rightarrow U$,
- Menge T der Transferfunktionen abgeschlossen unter Komposition
- Wirkung von Vorgängeranweisungen auf A modelliert durch Suprenum *sup* der Vorgängereigenschaften

10.2 Monotone Datenflußrahmen II

Lösung von Datenflußgleichungen ist Fixpunktberechnung

- monotoner Datenflußrahmen: (U,T),
- Anwendung eines monotonen Datenflußrahmens (U,T) gegeben durch Tripel $(G,F,\leq)$,
  - G: Ablaufgraph,
  - $F \subseteq T$: Menge anweisungsspezifischer monotoner Transferfunktionen
  - $\leq$: Ordnungsrelation mit $(U, \leq)$ vollständig halbgeordnet
- Tripel $(G,F,\leq)$, definiert eine CPO, oft einen Verband
- Datenflußgleichung df für Anweisungen ist stetig bzgl. dieser CPO:

$$E_{in}(A) = \sup_{X \in \text{pred}(A)}(E_{out}(X))$$
$$E_{out}(A) = f_A(E_{in}(A))$$

mit Transferfunktion $f_A \in F$ der Anweisung A

10.2 Vier DFA-Schemata

- Vorwärts und sicher:

$$E_{in}(A) = \sup_{X \in Vor(A)} E_{out}(X)$$

$$E_{out}(A) = f_A(E_{in}(A)) = E_{in}(A) - kill(A) \cup gen(A)$$

- rückwärts und sicher:

$$E_{out}(A) = \sup_{X \in Nach(A)} E_{in}(X)$$

$$E_{in}(A) = E_{out}(A) - kill(A) \cup gen(A)$$

- vorwärts und möglich:

$$E_{in}(A) = \bigcup_{X \in Vor(A)} E_{out}(X)$$

$$E_{out}(A) = E_{in}(A) - kill(A) \cup gen(A)$$

- rückwärts und möglich:

$$E_{out}(A) = \bigcup_{X \in Nach(A)} E_{in}(X)$$

$$E_{in}(A) = E_{out}(A) - kill(A) \cup gen(A)$$

$kill(A)$: Eigenschaften E, durch A beseitigt

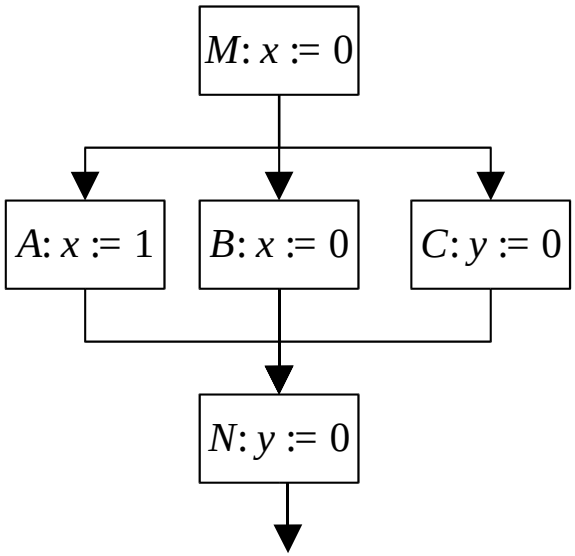
$gen(A)$: Eigenschaften E, durch A neu definiert (aufgabenspezifisch)

10.2 Monotone Datenflußrahmen III

- Gegeben:
  - $CPO\ (U, \leq)$ mit kleinstem Element $\perp$
  - Gerichteter (Steuerfluß-)Graph $G = (E, K)$ mit $|E| = k$ und Startecke e^0.
    $pred(e)$ sei Menge der Vorgängerknoten von e in Graph G
  - Transferfunktionen für Ecken $t_0 \ldots t_{k-1}$ mit $t_i: U \rightarrow U$ stetig bzgl. $\leq$
  - Initialisierung $\iota \in U$ für e^0
- Es gilt:
  - $((E \times U \times U)^k, \subseteq)$ ist eine CPO, wobei:
    $[(e^0, in^0, out^0) \ldots (e^{k-1}, in^{k-1}, out^{k-1})] \subseteq [(\underline{e^0}, \underline{in^0}, \underline{out^0}) \ldots (\underline{e^{k-1}}, \underline{in^{k-1}}, \underline{out^{k-1}})] \Leftrightarrow$
    $\forall 0 \leq i \leq k\text{-}1: e^i = \underline{e^i} \wedge in^i \leq \underline{in^i} \wedge out^i \leq \underline{out^i}$
  - Kleinstes Element ist Vektor $[(e^0, \iota, \perp), (e^1, \perp, \perp), \ldots, (e^{k-1}, \perp, \perp)]$
  - Sei $E_{in}(e, in, out) = in$ und $E_{out}(e, in, out_t) = u_{out}$
  - Funktion df: $(E \times U \times U)^k \rightarrow (E \times U \times U)^k$ mit
    $df\ ([(e^0, in^0, out^0) \ldots (e^{k-1}, in^{k-1}, out^{k-1})]) = [(e^0, \iota, t_0(\iota)) \ldots (e^{k-1}, \sup(PRE(e^{k-1})), t_{k-1}(in^{k-1}))]$
    wobei: $PRE(e) = \{out \mid out = E_{out}(e_i, in_i, out_i) \wedge e_i \in pred(e)\}$
  - Funktion df ist stetig bzgl. $\subseteq$

10.2 Beispiel I

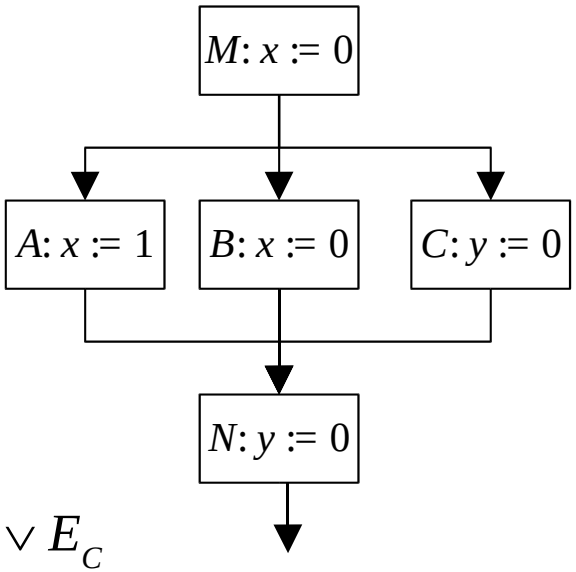
- Eigenschaft E: $x = 1$ garantiert?
- Wertebereich: boolescher Verband
- Transferfunktionen: *wahr*, *falsch*, *id*
  - Anweisung A: *wahr*
  - Anweisung B: *falsch*
  - Anweisung C: *id* d.h. unverändert



- Sind E_A , E_B , E_C die Werte von E nach den Anweisungen A,B,C, so gilt $\underline{E}_N = E_A \wedge E_B \wedge E_C$
- Vorwärtsanalyse
  - Beginn mit $\underline{E}_{A,B,C,N}$ = *wahr* vor den Anweisungen (Annahme $x = 1$)
  - Initialisierung $\underline{E}_M$ = *falsch* vor Anweisung M ist $x \neq 1$
  - Iteration liefert Fixpunkt E_N = *falsch*
- $x := 1\text{-}x$ wäre schwierig: Transferfunktion *not* nicht monoton

10.2 Beispiel II

- Eigenschaft E: $x = 1$ möglich?
- Wertebereich: boolescher Verband
- Transferfunktionen: *wahr*, *falsch*, *id*
  - Anweisung A: *wahr*
  - Anweisung B: *falsch*
  - Anweisung C: *id* d.h. unverändert
- Sind E_A , E_B , E_C die Werte von E nach den Anweisungen A,B,C, so gilt $\underline{E}_N = E_A \vee E_B \vee E_C$
- Vorwärtsanalyse
  - Beginn mit $\underline{E}_{A,B,C,N}$ = *falsch* vor den Anweisungen (Annahme $x \neq 1$)
  - Initialisierung $\underline{E}_M$ = *wahr* vor Anweisung M ist $x = 1$
  - Iteration liefert Fixpunkt E_N = *wahr*
- Verallgemeinertes Beispiel: berechne die Eigenschaften mehrerer Variabler gleichzeitig (Wertebereich Bitvektorverband)

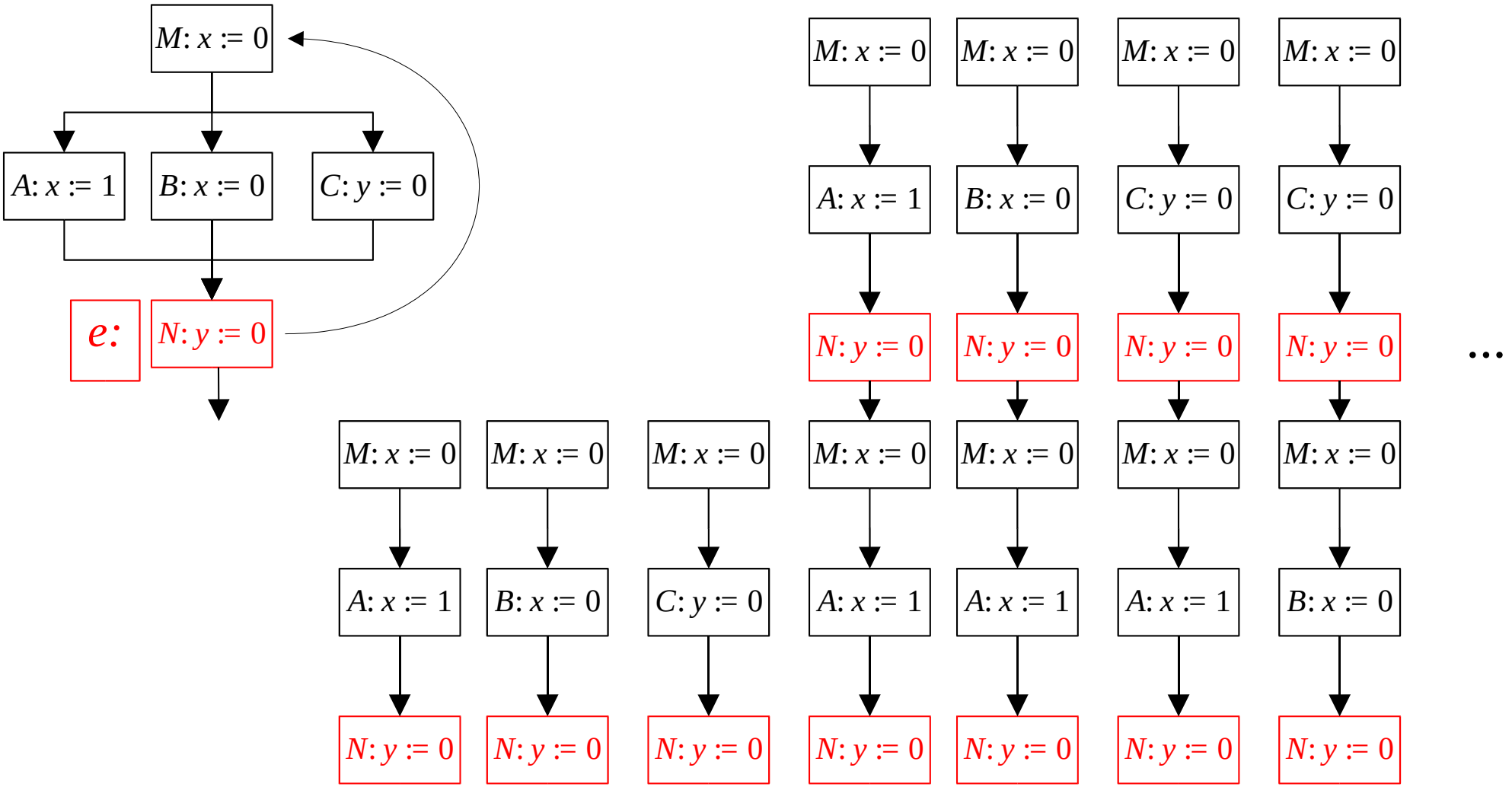


10.2 *MFP* und *MOP*

Bei stetigen Datenflußrahmen $DFG = (G,U,\iota,f)$:

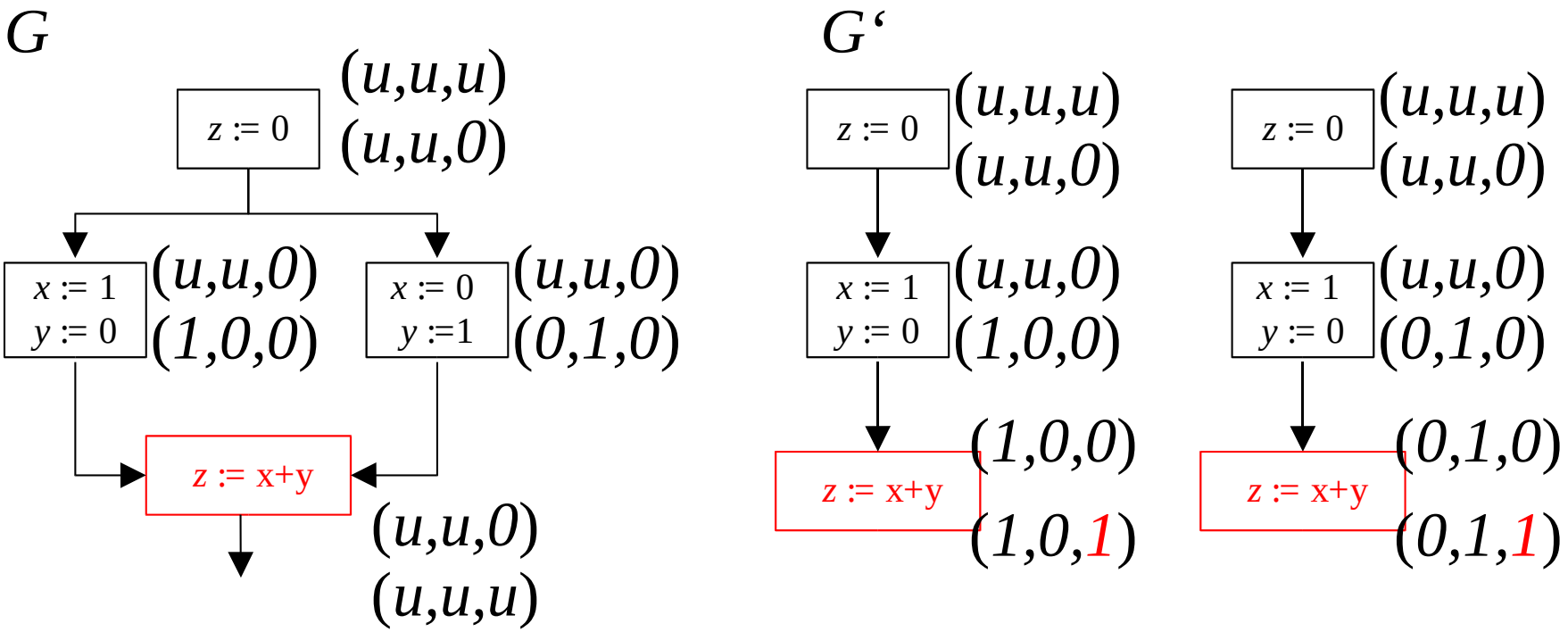
- minimaler Fixpunkt *MFP* durch iteratives Anwenden von f auf kleinstes El.
- Für alle Ecken $e \in E$ von $G = (E, K)$:
  - Pfadgraph $G'(e) = (E', K')$ mit
    - Für jeden Pfad P endend in e: $e' \in P \Leftrightarrow e' \in E'$
    - $(e', e'') \in K' \Leftrightarrow (e', e'') \in P$
    - azyklisch, unendlich, wenn G Schleifen enthält
  - Sei $DFG'(e) = (G'(e), U, \iota, f)$.
- Der *MOP* (*meet over all path*) von DFG in Ecke e ist der minimale Fixpunkt von $DFG'(e)$ in der Ecke e.
- *MFP* entspricht *MOP*, wenn f distributiv bzgl. *sup* in U, sonst *MFP* konservative Abschätzung des *MOP*
- Achtung: Tatsächliche Ausführbarkeit eines Pfades unentscheidbar, daher ist bereits *MOP* konservative Abschätzung des gewünschten Analyseergebnisses

10.2 Ablaufgraph G und Pfadgraph G'



10.2 Beispiel für $MFP(G) \neq MOP(G)$

Konstantenweitergabe: Wertevektor: $(x,y,z) \in \{0,1,unbekannt\}^3$



Kapitel 10: Datenflußanalyse

0. Einbettung
1. DFA-Schemata
2. Theorie
   - CPO, Fixpunktsatz
   - Monotone Datenflußrahmen
   - MFP & MPO
3. Konkrete Analysen
   - 3.1 Erreichende Definitionen
   - 3.2 Gültige Ausdrücke
4. Abstraktionen der Ablaufsteuerung
   - 4.1 Intervallanalyse
   - 4.2 Dominanzanalyse
   - 4.3 Eigenschaften der DG

10.3.1 Erreichende Definitionen (reaching definitions)

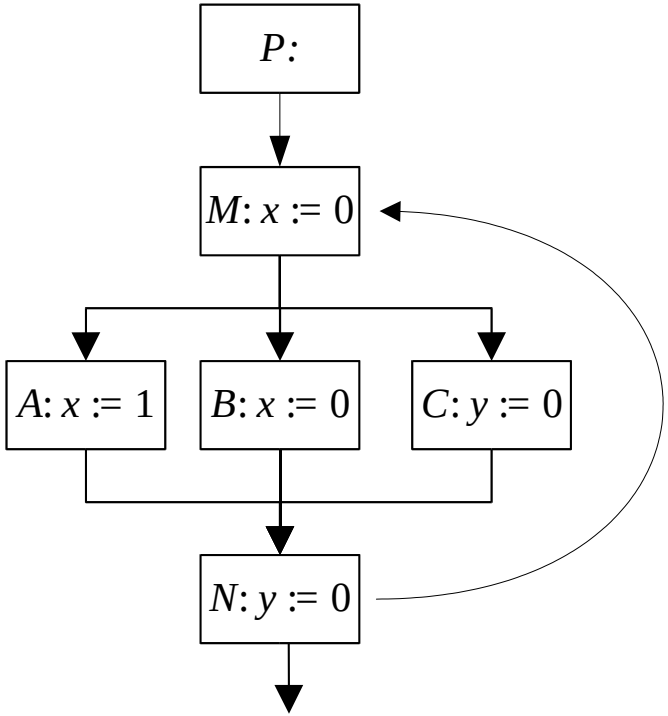
- Welche „Definitionen" (Zuweisungen) sind bei N gültig?
  - Für jede Variable und jede Ecke im Ablaufgraph höchstens eine Definition $\subseteq \{A_1 \ldots A_n\}$
  - Vorwärtsanalyse, sicher
  - Schema:

$$E_{in}(A) = \bigcap_{X \in Vor(A)} E_{out}(X)$$

$$E_{out}(A) = E_{in}(A) - kill(A) \cup gen(A)$$

- Initialisierung:
  - Definitionen aller Vorgänger erreichen A_i für alle Variablen
  - $E_{in}(A_1) = \emptyset, E_{in}(A_i) = \{\text{alle Variable}\}$ für alle anderen Ecken
- Eine Definition wird erzeugt ($gen(A)$) durch Zuweisung `x:=expr`
- eine Definition `x:=expr` wird ungültig ($kill\ (A)$) durch neue Definition `x:=expr´`

10.3.1 Beispiel



$$E_{out}(P) = E_{in}(P) = \varnothing$$

$$E_{in}(M) = E_{out}(P) \cap E_{out}(N) = \varnothing$$

$$E_{out}(M) = E_{in}(M) - \{x\} \cup \{x := 0\}$$

$$E_{in}(A) = E_{out}(M) = \{x := 0\}$$

$$E_{out}(A) = E_{in}(A) - \{x\} \cup \{x := 1\}$$

$$E_{in}(B) = E_{out}(M) = \{x := 0\}$$

$$E_{out}(B) = E_{in}(B) - \{x\} \cup \{x := 0\}$$

$$E_{in}(C) = E_{out}(M) = \{x := 0\}$$

$$E_{out}(C) = E_{in}(C) - \{y\} \cup \{y := 0\}$$

$$E_{in}(N) = E_{out}(A) \cap E_{out}(B) \cap E_{out}(C) = \varnothing$$

$$E_{out}(N) = E_{in}(N) - \{y\} \cup \{y := 0\}$$

10.3.2 Gültige Ausdrücke (available expressions)

- Welche „Ausdrücke" (Werte) sind bei N gültig?
- Für jede Ecke höchstens eine Definition $\subseteq \{e_1 \ldots e_N\}$
  (Menge der (Teil-)Ausdrücke e_i)
- Vorwärtsanalyse, sicher
- Schema:

$$E_{in}(A) = \bigcap_{X \in Vor(A)} E_{out}(A)$$

$$E_{out}(A) = E_{in}(A) - kill(A) \cup gen(A)$$

- Initialisierung:
  - Alle Ausdrücke in allen Ecken gültig ($\{e_1 \ldots e_N\}$) für alle Variablen
  - jedoch $E_{in}(A_1) = \varnothing$
- ein Teilausdruck wird erzeugt ($gen(A)$) durch Berechnung expr
- ein expr wird ungültig ($kill\ (A)$) durch Definition x:=expr´, wenn x Blatt (Operand) im Ausdrucksbaum von expr ist
- sonst wie erreichende Definitionen

Kapitel 10: Datenflußanalyse

0. Einbettung
1. DFA-Schemata
2. Theorie
   - CPO, Fixpunktsatz
   - Monotone Datenflußrahmen
   - MFP & MPO
3. Konkrete Analysen
   - 3.1 Erreichende Definitionen
   - 3.2 Gültige Ausdrücke
4. Abstraktionen der Ablaufsteuerung
   - 4.1 Intervallanalyse
   - 4.2 Dominanzanalyse
   - 4.3 Eigenschaften der DG

10.4 Abstraktionen des Ablaufgraphen

Aufrufgraph: Ecken - Prozeduren; Kanten - Aufrufe (ohne Rücksicht auf Aufrufreihenfolge oder Parameter)

Verfeinerung: Reihenfolge oder Parameter berücksichtigen

Vorsicht: Aufrufgraph erlaubt Wege

Aufruf $\rightarrow$ Prozedur $\rightarrow$ Rückkehr zu anderem Aufruf

Ablaufgraph einer Prozedur:

- Ecken: - Anweisungen (Ausdrücke)
- Kanten - (essentielle) Abhängigkeiten zwischen Zuweisungen (Ausdrücken)
- Ziele: Anwendung von monotonen Datenflußrahmen
- Effizientere Berechnung des MFP, wenn Berechnungsreihenfolge der Schleifenschachtelung entspricht
- Schleifen-Schachtelung aus Ablaufgraph berechnen

10.4.1 Intervalle und Intervallanalyse

Intervall $I(n)$ einer Ecke n in einem Ablaufgraph G:

n_0 Startecke, vor Vorgängerrelation in G:

1. $\mathrm{n} \in I(n)$
2. $\forall m, m \neq n_0 : (\forall p : p \; vor \; m \wedge p \in I(n)) \Rightarrow m \in I(n)$
3. $I(n)$ enthält keine weiteren Ecken

Intervalle von G partitionieren Eckenmenge

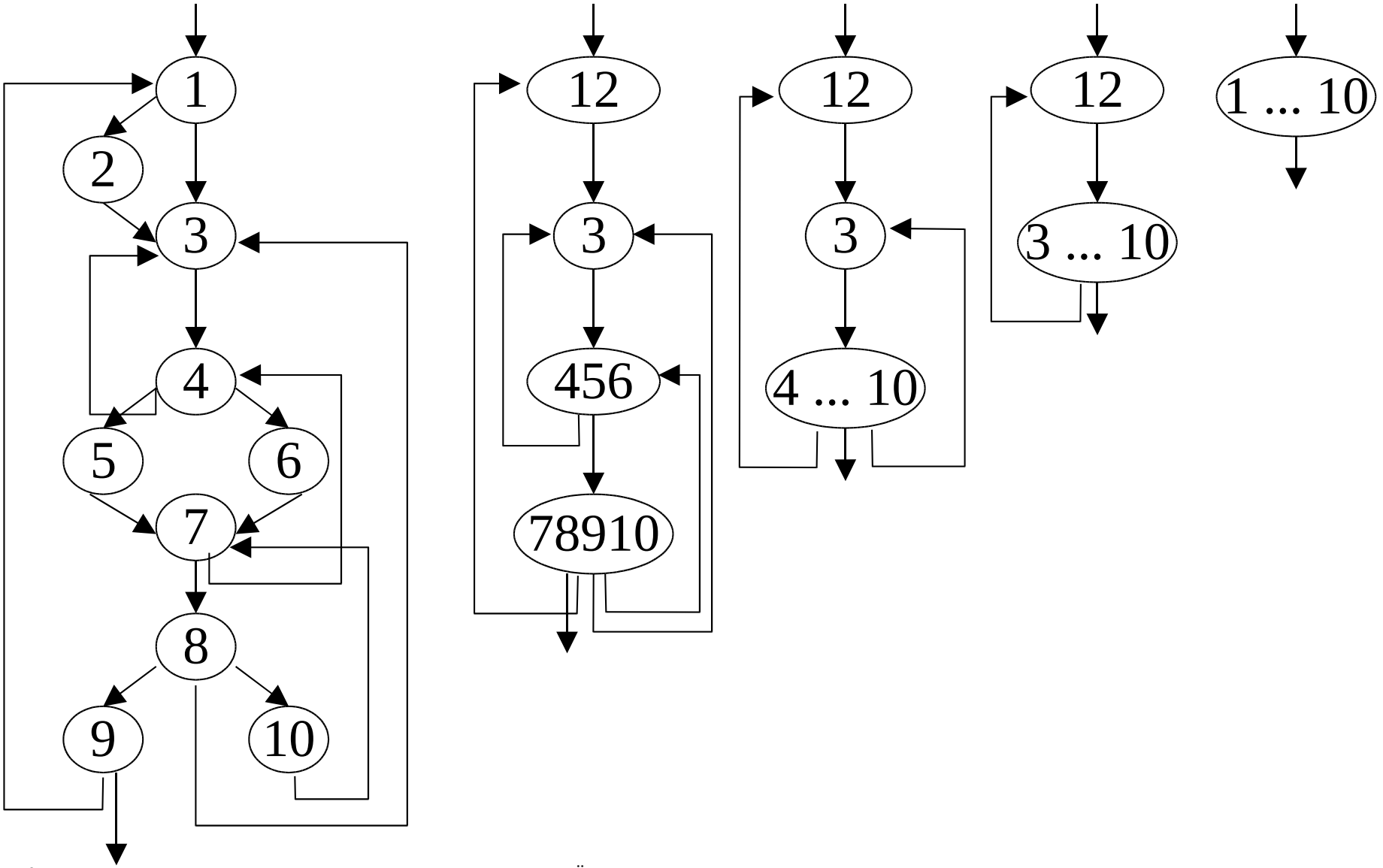
Intervallgraph $I(G)$:

1. Ecken in $I(G)$ sind Intervalle
2. Startecke ist Intervall $\mathrm{I}(\mathrm{n}_0)$
3. Kante (N, M) in $I(G)$ gdw. $\exists n \in \mathrm{N} \wedge m \in M \wedge (n, m)$ in G

Intervallanalyse: Berechne iteriert Intervallgraphen
$I(G), I(I(G)), I(I(I(G))), \ldots$

Intervallanalyse zieht schrittweise Schleifen zu Ecken zusammen

10.4.1 Beispiel Intervallanalyse



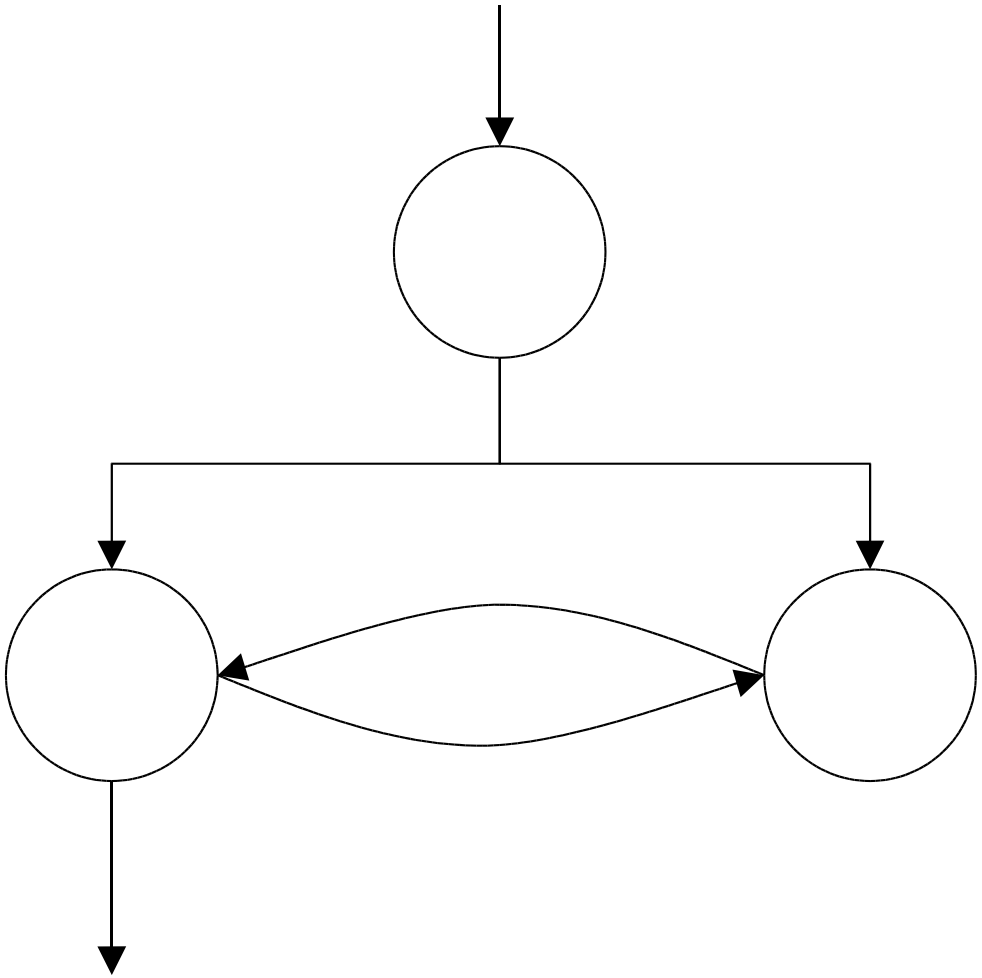
10.4.1 Reduzible Ablaufgraphen

- Ablaufgraph reduzibel $\Leftrightarrow$ Intervallanalyse liefert einzelne Ecke
- goto-freie Sprachen liefern immer reduzible Ablaufgraphen
- Reduzibilität auch durch T_1 - T_2 Analyse feststellbar:

  T_1 : Reflexive Kanten löschen

  T_2 : Ecken mit nur einem Vorgänger mit diesem verschmelzen
- T_1 - T_2 Analyse ist konfluent, d. h. Anwendung der Transformationen in beliebiger Reihenfolge möglich

10.4.1 Beispiel nicht-reduzibler Ablaufgraph



10.4.2 Dominanz und Dominatorbäume

Dominanz: $X \geq Y$

Auf jedem Pfad von der Startecke S im Ablaufgraph kommt X vor Y.

$\geq$ ist reflexiv: $X \geq X$.

Strikte Dominanz:

$$X > Y \quad \Leftrightarrow \quad X \geq Y \wedge X \neq Y.$$

Unmittelbare (direkte) Dominanz: $ddom(X)$

$$X = ddom(Y) \quad \Leftrightarrow \quad X > Y \ \wedge \neg \exists\, Z\colon X > Z > Y.$$

Nach-Dominanz: $X \leq Y$

Auf jedem Pfad von Y zur Endecke E im Ablaufgraph kommt Y vor X.

Übrige Definitionen für Nach-Dominanz analog.

10.4.2 Dominanzanalyse

Vorwärtsanalyse für sichere Programmeigenschaften.

Gegeben Ablaufgraph $G = (E, K)$ mit Vor Vorgängerrelation in G:

$$D_{in}(n) = \bigcap_{p \in Vor(n)} D_{out}(p)$$
$$D_{out}(n) = D_{in}(n) \cup \{n\}$$

Vorbesetzung für Ecken:

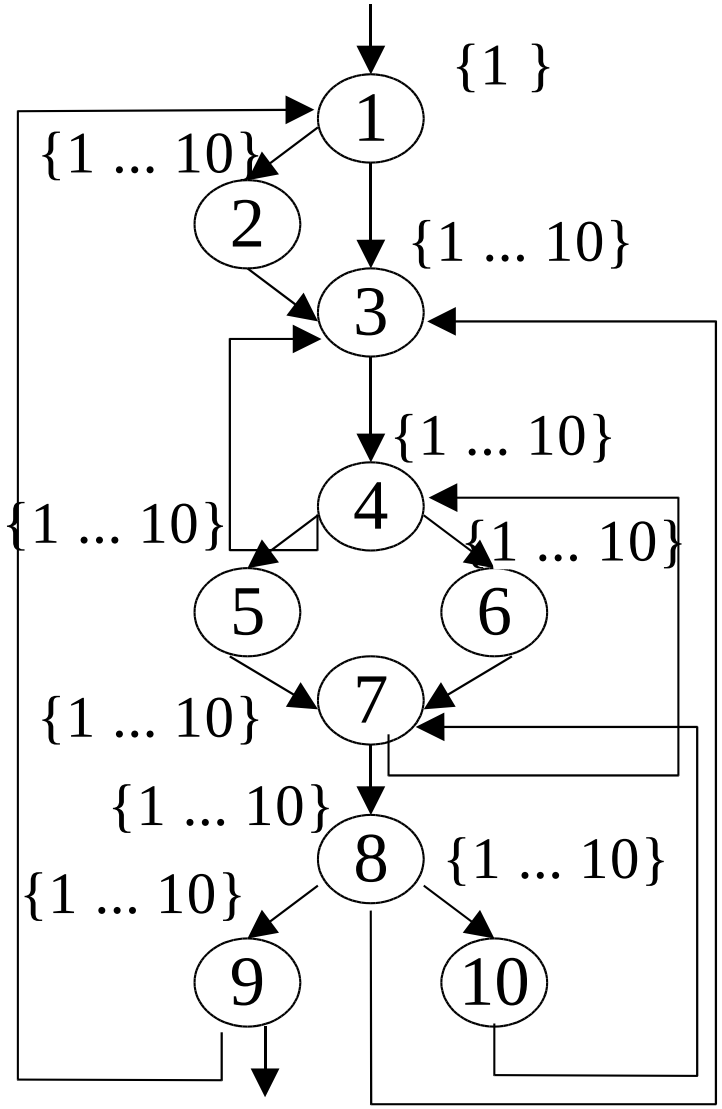
$Dout\,(n) = E$ (Annahme alle Ecken dominieren einander)

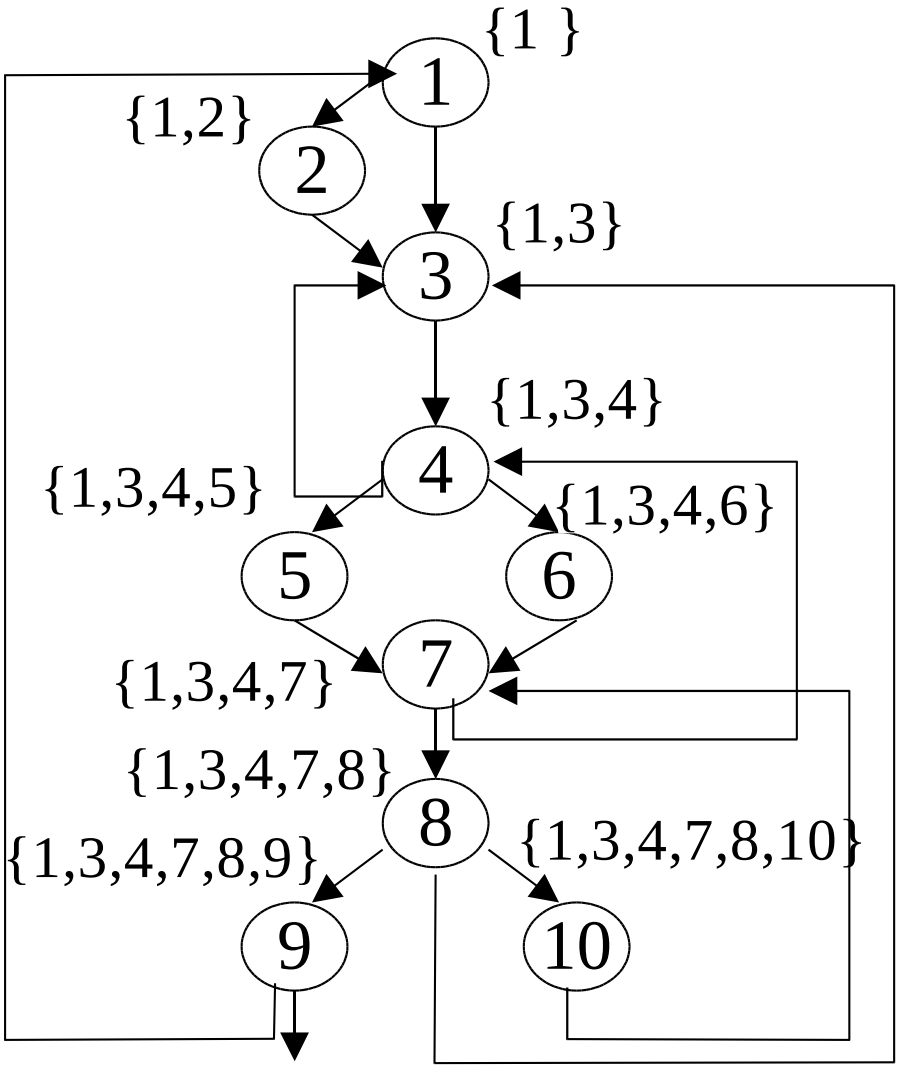
$Vor(n)$ nicht definiert für Startecke n_0 : $D_{in}(n_0) = D_{out}(n_0) = \{n_0\}$.

Vereinfachung:

- $D\,(n) = \{n\} \cup \bigcap_{p \in Vor(n)} D\,(p)$
- möglich weil in n keine Information ungültig wird
- mit Vorbesetzung $\forall n \in E - \{n_0\}: D(n) = E$ und $D(n_0) = \{n_0\}$

10.4.2 Beispiel

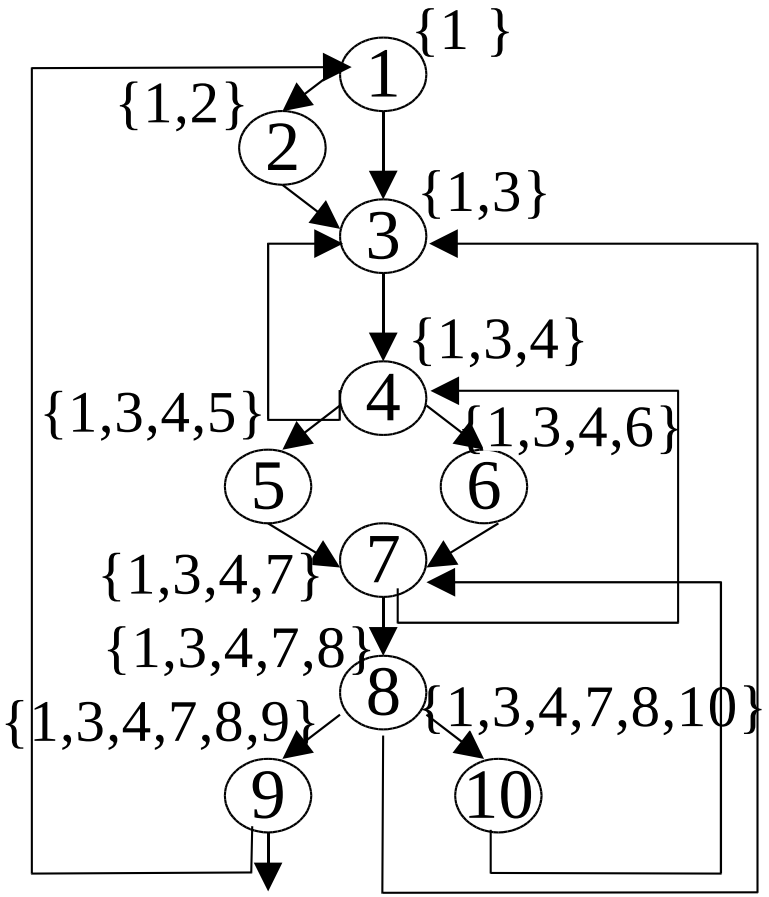


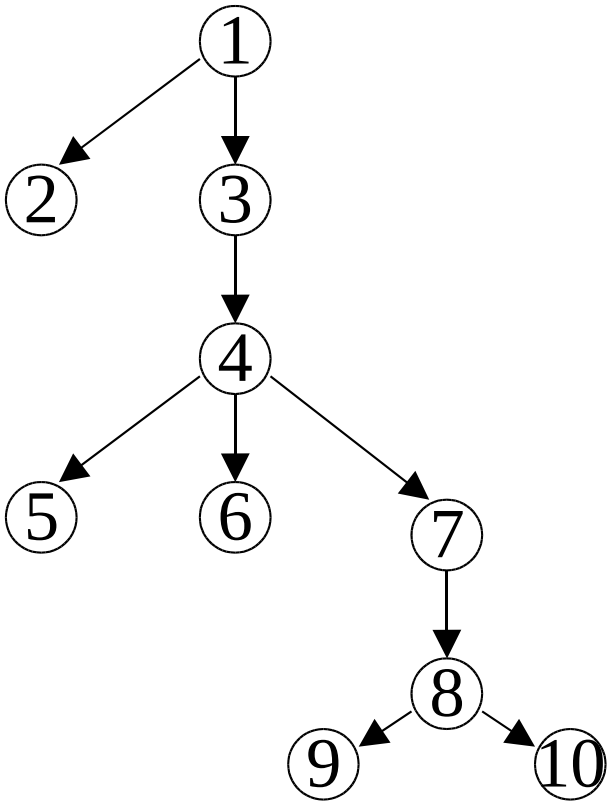


10.4.2 Dominatorbaum G_B

Dominatorbaum $G_D\,(G) = (E_D, K_D)$ eines Ablaufgraphen

$G = (E, K)$ mit $E_D = E$ und $(e_D, e'_D) \in K_D \Leftrightarrow e = ddom\,(e')$





10.4.2 Dominanzgrenze und iterierte Dominanzgrenze

Dominanzgrenze $DG\,(X)$

- Menge von Ecken die gerade nicht mehr von X dominiert werden
- $DG\,(X) = \{Y \mid \exists P \in Vor\,(Y): X \geq P \wedge \neg(X > Y)\}$.

Dominanzgrenze einer **Menge** M von Blöcken $DG\,(M)$

- $DG\,(M) = \bigcup_{X \in M} DG\,(X)$

Iterierte Dominanzgrenze $DG+\,(M)$

- minimaler Fixpunkt von:
  $DG_0 = DG\,(M)$,
  $DG_{i+1} = DG\,(M \cup DG_i)$

10.4.2 Berechnung der Dominanzgrenze

- Einfacher Algorithmus in $O(n^2)$

  (*a*) Sei $dfn(X)$ Nummer von X in Tiefensuchordnung über dem Ablaufgraph. Es gilt:

  $$Z \in DG(X) \Rightarrow dfn(ddom(X)) \geq dfn(ddom(X))$$

  (*b*) Sei $SSC(Z)$ starke Zusammenhangskomponente von Z im Ablaufgraph und (per Definition)

  $$equidom(Z) = \{ Y \mid ddom(Z) = ddom(Y) \}$$

  Es gilt:

  $$(\forall Y \in equidom(Z) : Y \in SSC(Z)) \Rightarrow (Z \in DG^+(X) \Rightarrow equidom(Z) \in DG^+(X))$$

- Ausnutzung dieser Eigenschaften:

  (*a*) Besuchsreihenfolge in umgekehrter Tiefensuche der direkten Dominatoren

  (b) Gemeinsame Behandlung von stark zusammenhängenden Ecken

- führt zu Algorithmus in (fast) $O(n^2)$

10.4.3 Eigenschaften unmittelbarer Dominatoren

$$Z \in DG(X) \Rightarrow dfn(ddom(X)) \geq dfn(ddom(X))$$

(1) $(Y, Z) \in K \Rightarrow ddom(Z) \geq Y$

(2) $(Y, Z) \in K \wedge Y \neq ddom(Z) \Rightarrow ddom(Z) \geq ddom(Y) > Y$

(3) X vor $ddom(Z)$ im Tiefensuchbaum $\wedge\ ddom(Z) \geq Y \Rightarrow X \geq Y \Rightarrow X \geq Z$

(4) X vor $ddom(Z)$ im Tiefensuchbaum $\Rightarrow Z \notin DG(X)$

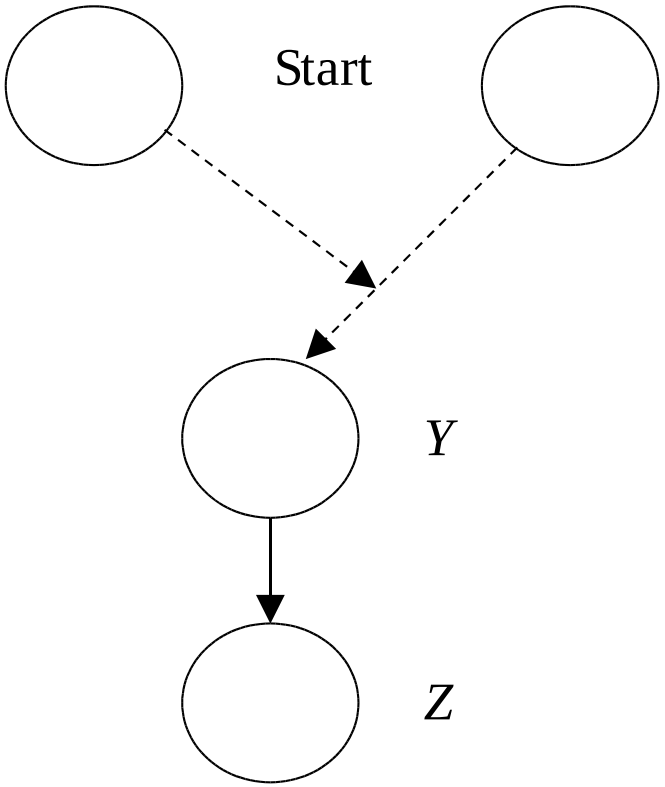
(5) $Z \in DG(X) \Rightarrow dfn(X) > dfn(ddom(Z))$

(6) $Z \in DG(X) \Rightarrow ddom(Z) \geq X$

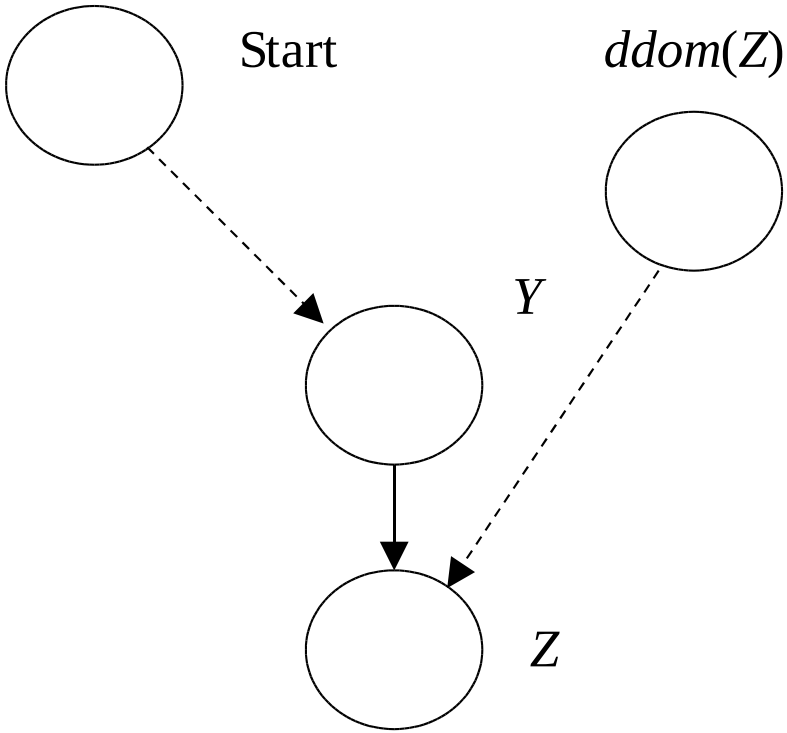
10.4.3 Eigenschaft (1)

$(Y, Z) \in K \Rightarrow ddom(Z) \geq Y$

- Gegenteil ergibt Widerspruch: $ddom(Z)$ kein Dominator.



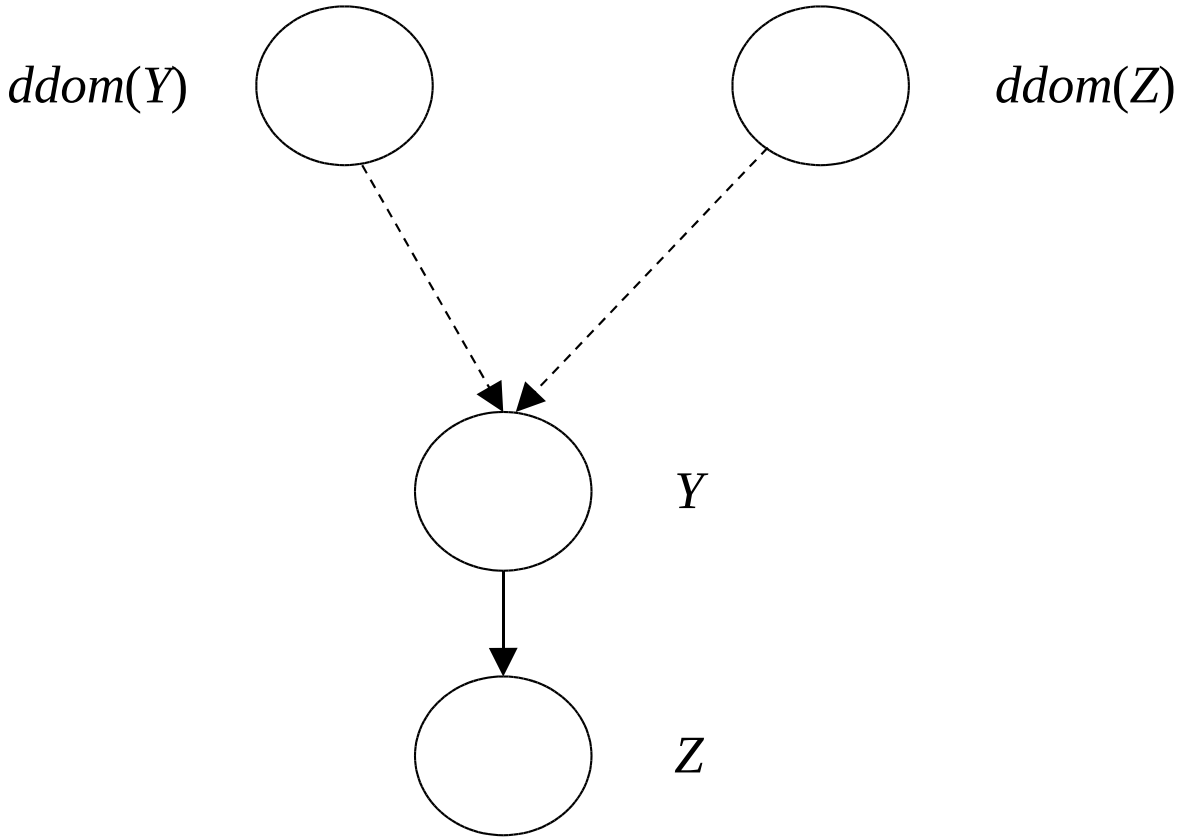
$ddom(Z)$



10.4.3 Eigenschaft (2)

$(Y, Z) \in K \wedge Y \neq ddom\,(Z) \Rightarrow ddom\,(Z) \geq ddom\,(Y) > Y$

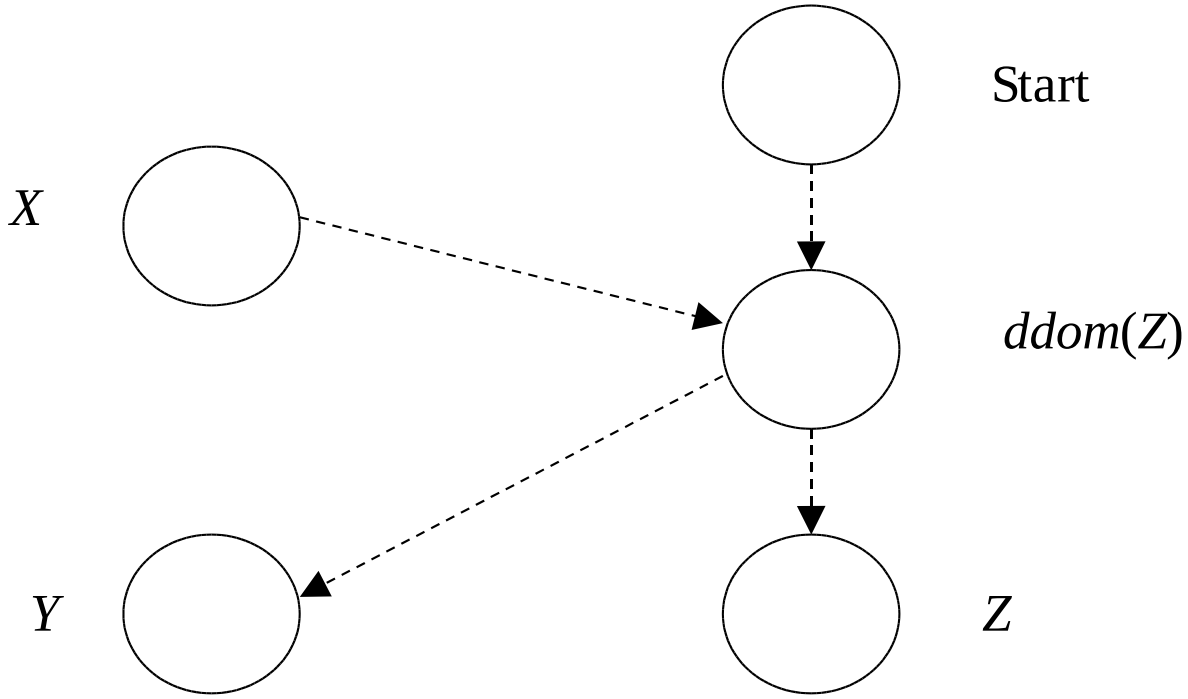
- Gegenteil ergibt Widerspruch: $ddom(Y)$ kein Dominator.



10.4.3 Eigenschaft (3)

X vor $ddom$ (Z) im Tiefensuchbaum $\wedge$ $ddom$ $(Z) \geq Y \Rightarrow X \geq Y \Rightarrow X \geq Z$

- Gegenteil ergibt Widerspruch: $X \geq Y$



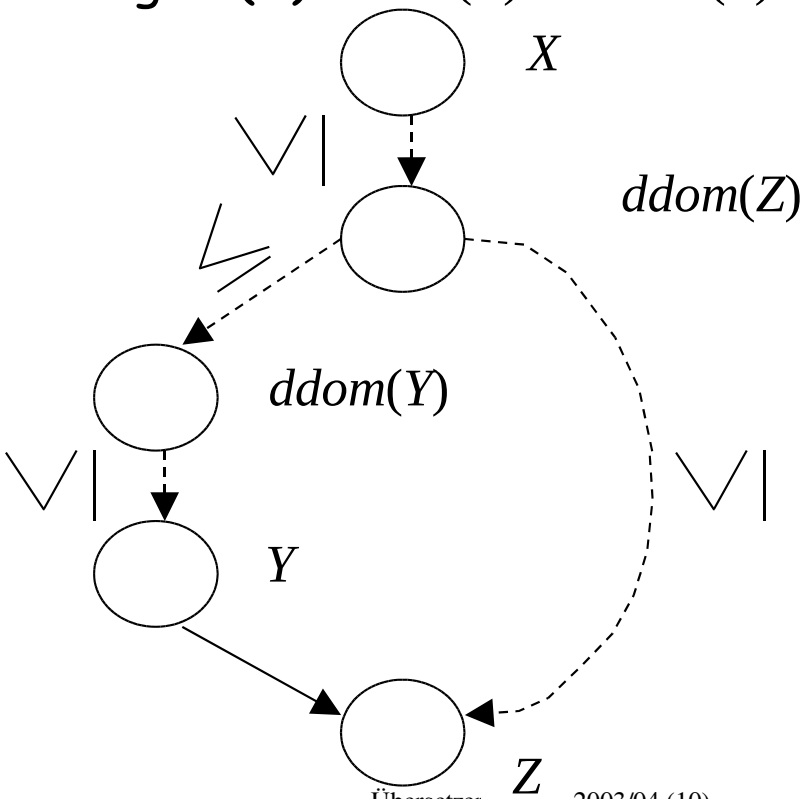
10.4.3 Eigenschaft (4)

X vor $ddom\ (Z)$ im Tiefensuchbaum $\Rightarrow Z \notin DG\ (X)$

- Gegenteil ergibt Widerspruch: $\exists\ (Y, Z)$: $X \geq Y \wedge X > Z$

  (a) $ddom\ (Z) = Y \Rightarrow X \geq Z$

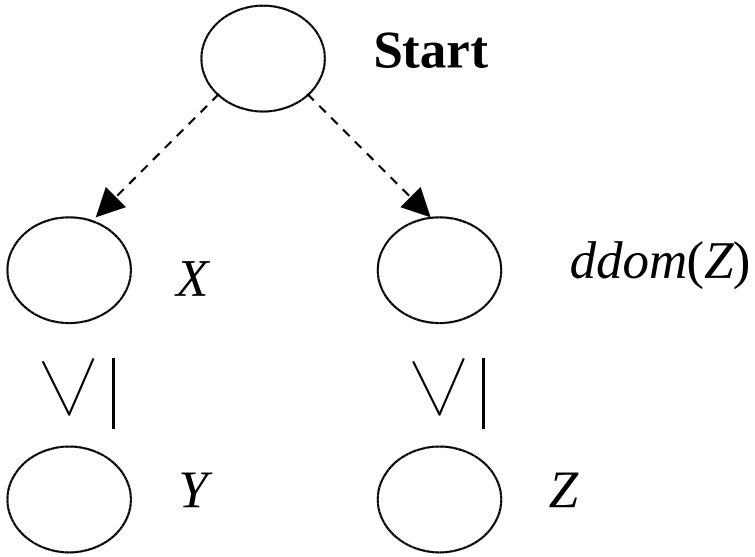
  (b) $ddom\ (Z) \neq Y$ wegen (2) $ddom\ (Z) \geq ddom\ (Y)$ und nach (3) $X \geq Y \Rightarrow X \geq Z$.



10.4.3 Eigenschaft (5)

$Z \in DG(X) \Rightarrow dfn(X) > dfn(ddom(Z))$

- Gegenteil ergibt jeweils Widerspruch:
  (a) X vor $ddom(Z)$ im Tiefensuchbaum nach (4)
  (b) X „links von" $ddom(Z)$ im Tiefensuchbaum $\Rightarrow \nexists(Y, Z)$



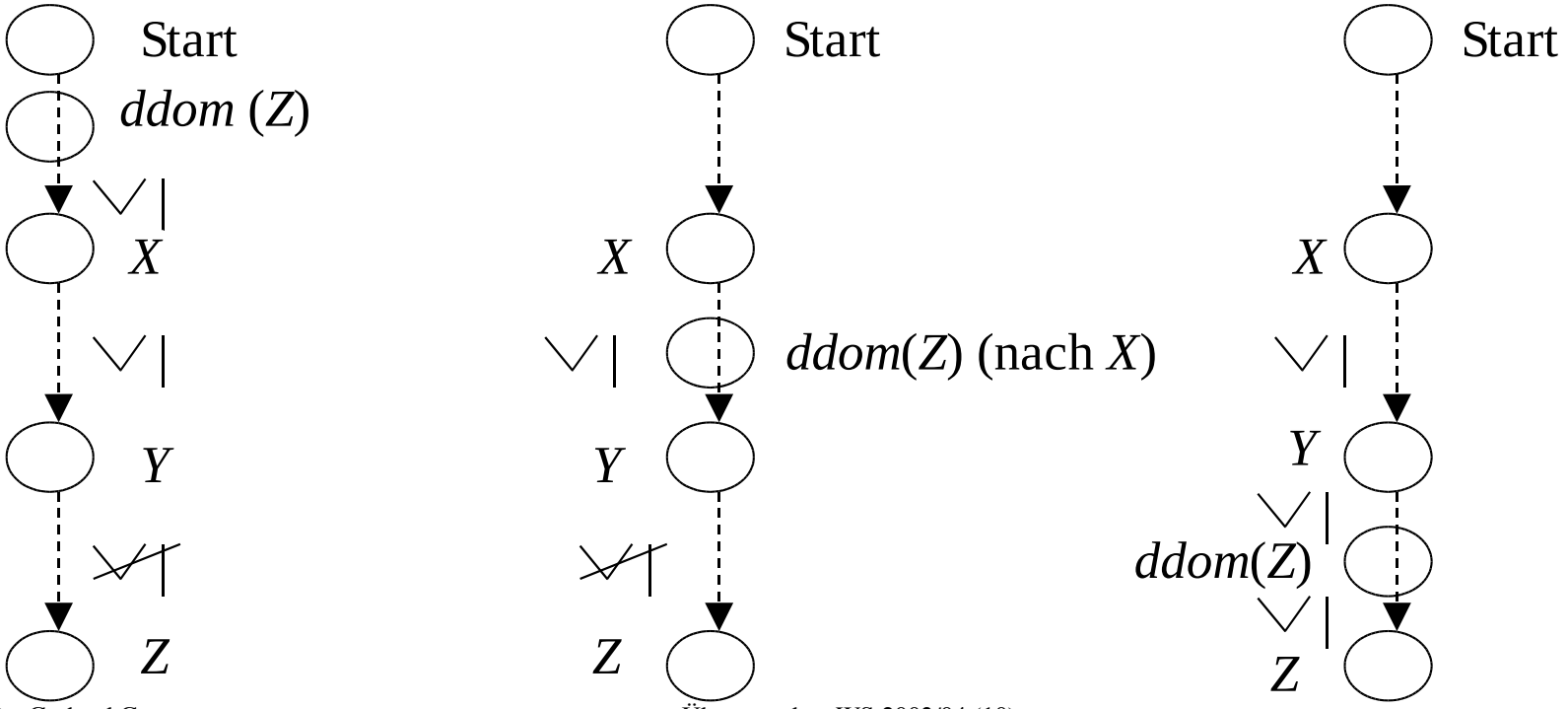
10.4.3 Eigenschaft (6)

$Z \in DG(X) \Rightarrow ddom(Z) \geq X$

- Gegenteil ergibt jeweils Widerspruch:

  (a) $X = ddom(Z) \Rightarrow Z \notin DG(X)$

  (b) $X \neq ddom(Z) \wedge ddom(Z) \not\geq X$



10.4.3 Eigenschaft

$$\forall Y \in equidom\,(Z) : Y \in SSC\,(Z) \Rightarrow Z \in DG^{+}\,(X) \Rightarrow equidom\,(Z) \subseteq DG\,(X)$$