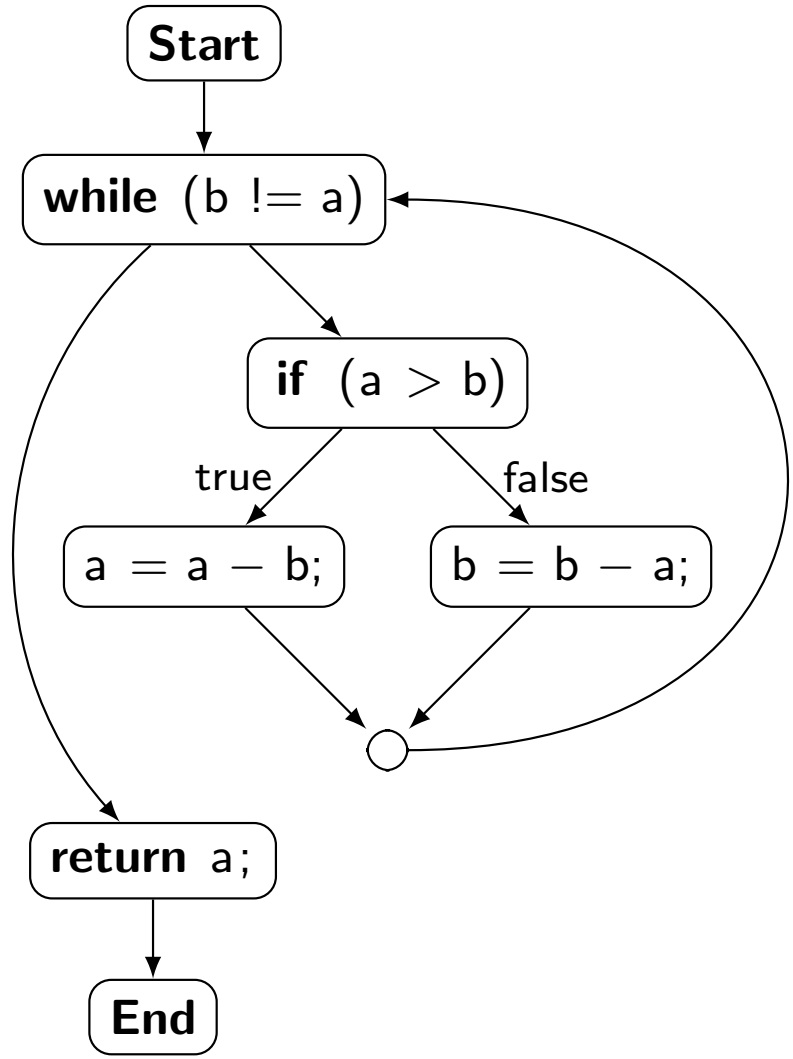
Dominanz

Wiederholung: Ablaufgraph (CFG)

- Jeder Grundblock besitzt einen entsprechenden Knoten im Ablaufgraph.
- Die Kanten zwischen den Knoten repräsentieren mögliche Sprünge zwischen den Grundblöcken.
- Es existieren 2 zusätzliche Knoten **Start** und **End** (auch Startblock und Endblock genannt).
- Es gibt eine Kante von **Start** zu jedem Grundblock mit dem das Programm betreten werden kann.
- Es gibt eine Kante von jedem Grundblock mit dem das Programm verlassen werden kann zu **End**.

Beispiel Ablaufgraph

```
int gcd(int a, int b)
{
    while(b != a) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return a;
}
```



Korrektheitsbedingung Ablaufgraph

$\forall$ Eingabe z: Programm durchläuft bei Eingabe z dynamisch die Folge von Grundblöcken

$$W = (\textbf{Start}, b_1, b_2, \ldots, b_n, \textbf{End})$$

$$\Longrightarrow \; W \text{ ist ein Pfad im CFG.}$$

Beachte: Die umgekehrte Implikation muss nicht gelten. Dh CFG darf zuviele Pfade enthalten (zB toter Code), aber niemals zuwenig! CFG ist *konservative Approximation*. Natürlich will man, dass CFG möglichst wenig nicht dynamisch realisierbare Pfade enthält (*Präzision*). CFGs für strukturierten Kontrollfluss (nur 1 Ein-Ausgang je Konstrukt, zB IF, WHILE, ...) sind i.a. präzise.

Dominanz

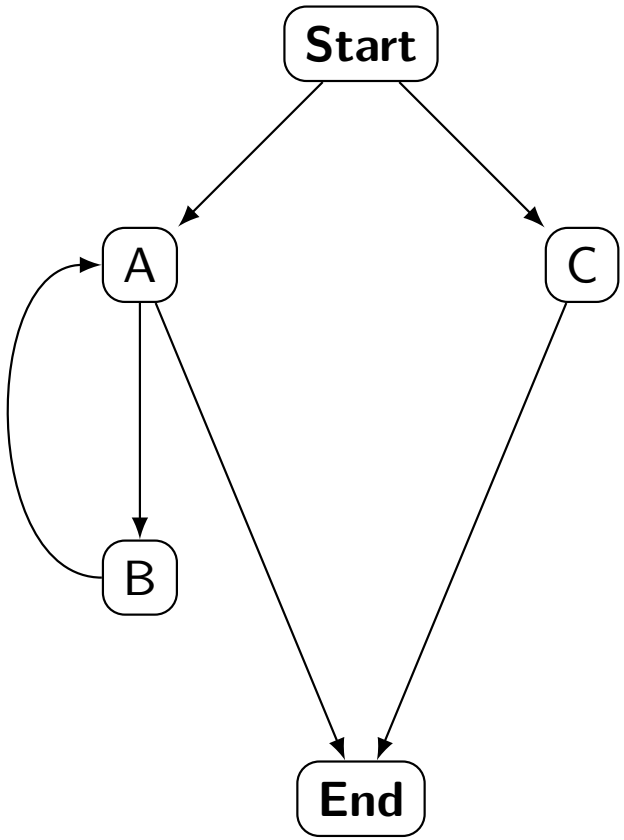
Definition Dominanz

X *dominiert* Y, wenn jeder Pfad im Ablaufgraph von **Start** zu Y auch X enthält. Wir schreiben $X \preceq Y$ (oft auch X dom Y)

Definition Postdominanz

Y *postdominiert* X, wenn jeder Pfad im Ablaufgraph von X zu **End** auch Y enthält. Wir schreiben Y postdom X.

Dominanz Beispiel



- **Start** $\preceq B$, $A \preceq B$
- $\neg(A \preceq C)$, $\neg(A \preceq$ **End**$)$
- **End** postdom B,
  End postdom C,
  A postdom B

Dominanz Anwendungen

- Schleifeneintrittspunkt $\preceq$ Schleifenrumpf
- Schleifenaustrittspunkt postdom Schleifenrumpf
- $X \preceq Y \wedge Y$ postdom $X \implies X, Y$ werden stets zusammen ausgeführt
- Kontrollflussregion: Bereich zwischen X und Y
- strukturierte Programme: Konstrukte (z.B. Schleifen) haben genau einen Eintritts- und einen Austrittspunkt
- Dominatorbaum: liefert Schachtelung von Schleifen, If-Anweisungen, u.ä. $\leftrightarrow$ "*Intervallanalyse*" [Tarjan]

Dominanz Eigenschaften

- Dominanz ist reflexiv: $X \preceq X$
- Dominanz ist transitiv: $X \preceq Y \wedge Y \preceq Z \Rightarrow X \preceq Z$.
- *strikte Dominanz*: $X \prec Y := X \preceq Y \wedge X \neq Y$.
- *direkte Dominanz* (englisch immediate dominator):

$$X = \text{idom}(Y) := X \prec Y \wedge \neg\exists Z : X \prec Z \prec Y$$

- Jeder Block außer dem Startblock hat genau einen direkten Dominator $\Rightarrow$ direkte Dominatoren bilden einen Baum, den *Dominatorbaum*.
  Dominatorbaum ist spezieller aufspannender Baum.
- Postdominanz ist ebenfalls reflexiv und transitiv. Definition von *strikter Postdominanz*, *direkter Postdominanz* und dem *Postdominanzbaum* analog.

GOTO-Elimination

Satz(Böhm, Jacopini 1964): Jedes Programm kann nur durch While / If-Then-Else / Begin-End dargestellt werden. [1]
$\Rightarrow$ Ersatz von GOTO durch strukturierte Schleifen.

Die so entstehenden Programme sind jedoch softwaretechnisch pervers

[1] Beweisidee: Durchnummerieren der Grundblöcke im CFG, "Program Counter"

Beispiel

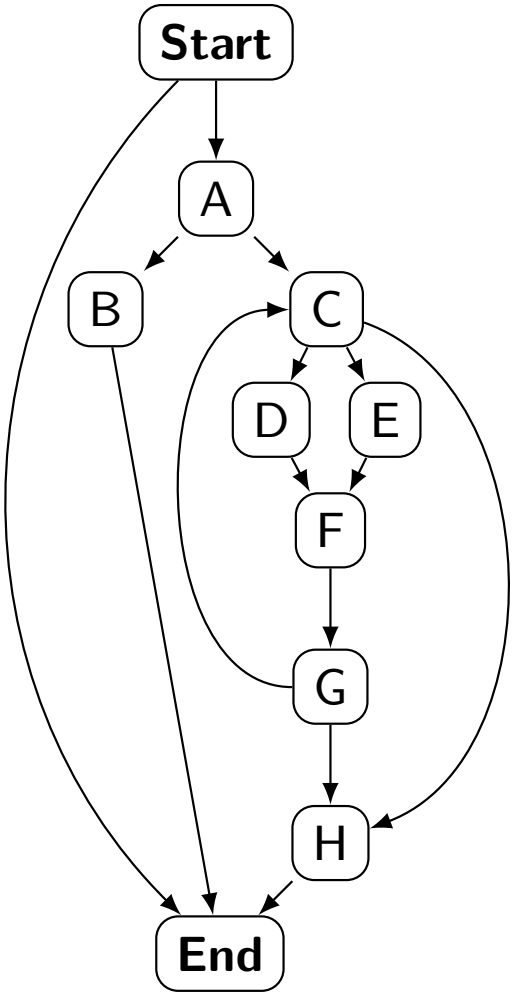


Abbildung: Ablaufgraph

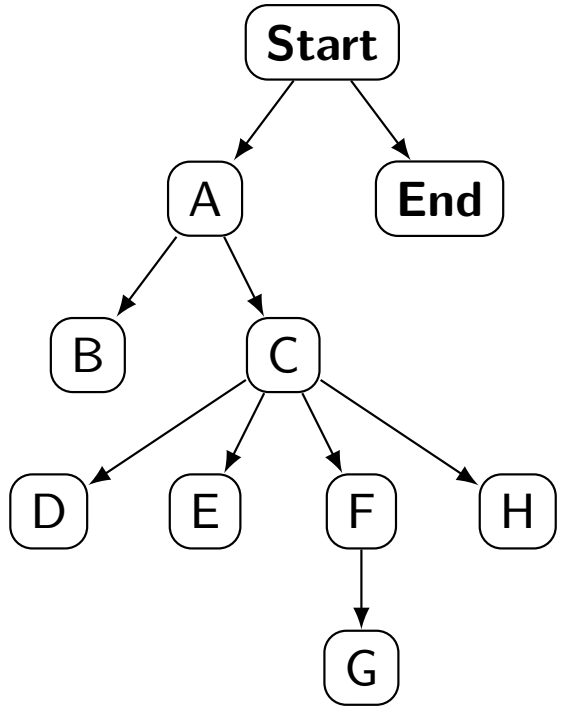


Abbildung: Dominanzbaum

GOTO-Elimination - Identifikation geschachtelter Schleifen

1. Dominatoren, zu denen Rückwärtskanten führen bilden Schleifeneintrittspunkte.
2. Postdominatoren der Eintrittspunkte bilden Schleifenaustrittspunkte.
3. Schleifen werden von innen nach aussen identifiziert (bottom-up Durchlauf im Dominatorbaum).
4. Falls 1./2. nicht möglich so entspricht die Schleife einer *starken Zusammenhangskomponente*.
5. Kombination von Dominatoren und Zusammenhangskomponenten: finde zuerst strukturierte Schleifen.
6. Evtl. aufbrechen des Graphen und Kopieren von Knoten / Einfügen zusätzlicher Kontrollflags.

GOTO-Eliminiation - Beispiel

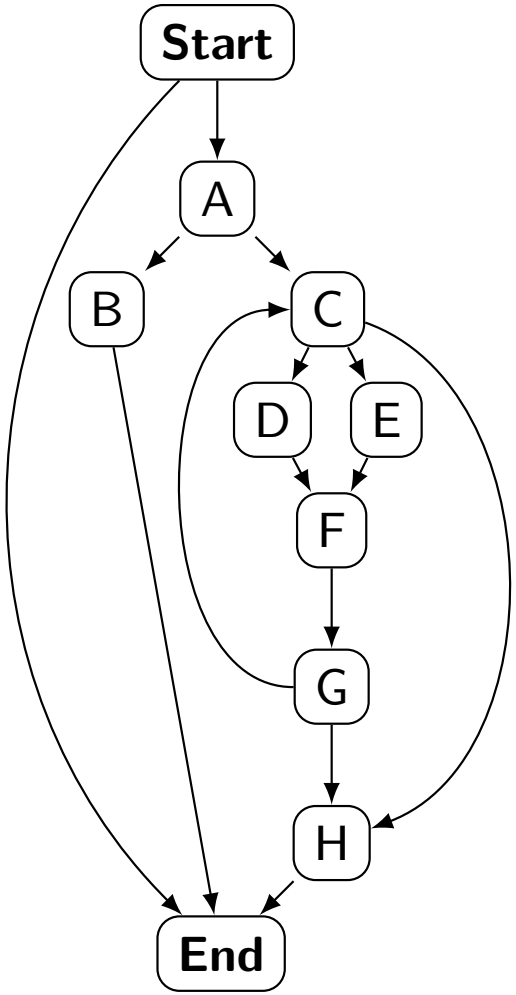


Abbildung: Ablaufgraph

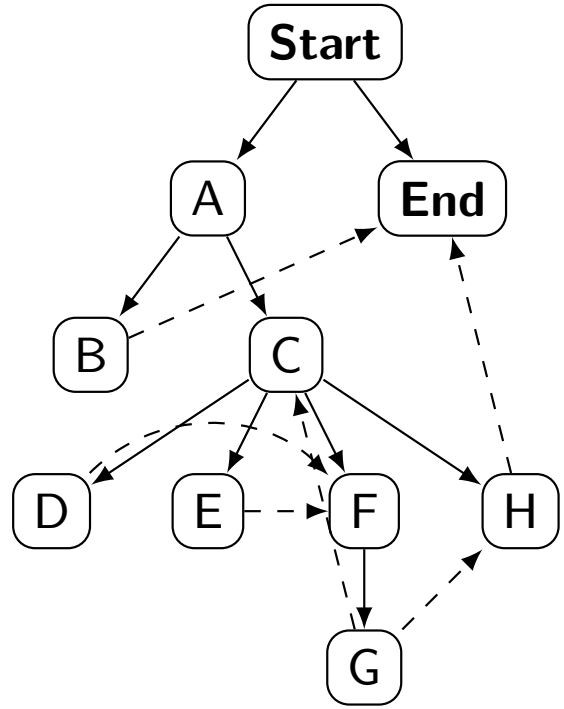
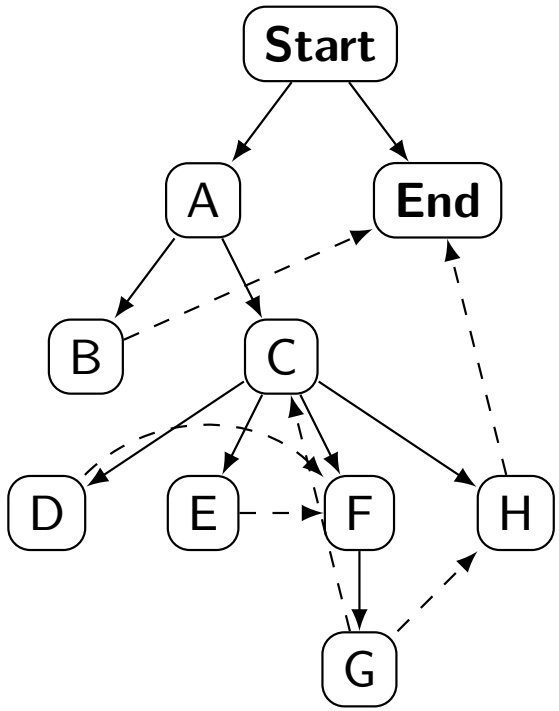


Abbildung: Dominanzbaum mit Ablaufkanten

GOTO-Eliminiation - Beispiel

```
if ( A ) {
    B;
} else {
    while( C1 && !G ) {
        if ( C2 ) {
            D;
        } else {
            E;
        }
        F;
    }
    H;
}
```



GOTO-Eliminiation - Beispiel

weiteres Beispiel: Einfügen der Kante (B, D)

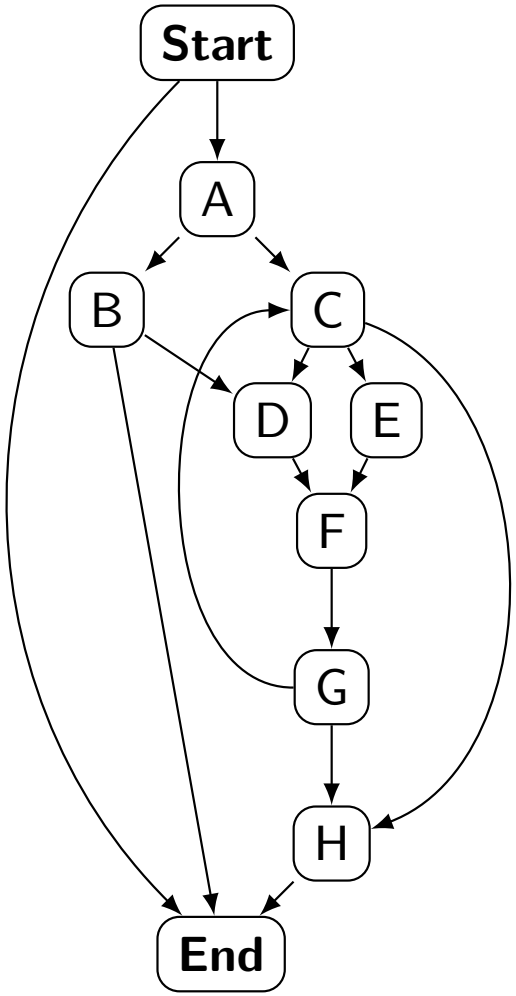


Abbildung: Ablaufgraph

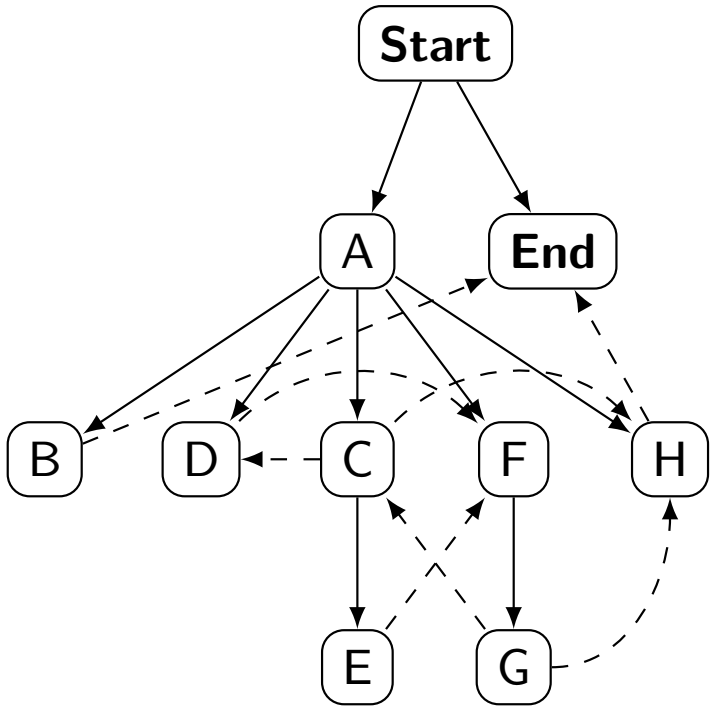
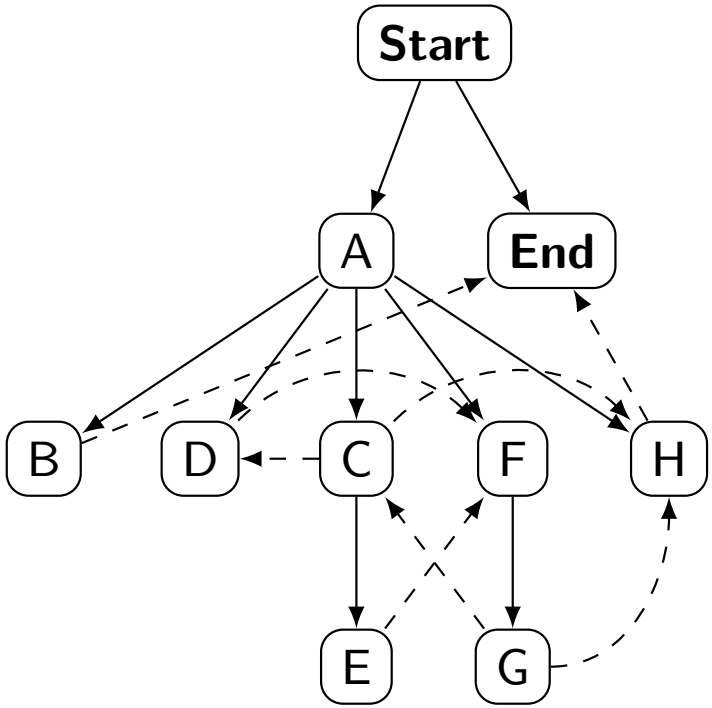


Abbildung: Dominanzbaum mit Ablaufkanten

GOTO-Eliminiation - Beispiel

```
if ( A ) {
    flag1 = true;
    if ( B ) { flag2 = true; }
    else     { flag2 = false; }
} else { flag1 = false; }

if ( !flag1 || flag2 ) {
    while ( C1 && !G || flag2) {
        if ( C2 || flag2 ) {
            D; flag2 = false;
        } else { E; }
        F;
    }
    H;
}
```



Berechnung mit Fixpunktiteration

Die Menge aller Dominatoren von X lässt sich darstellen als

$$\mathrm{dom}(X) = \{X\} \cup \bigcap_{Y \in \mathrm{pred}(X)} \mathrm{dom}(Y)$$

$\Longrightarrow$ Berechnung mit Fixpunktiteration möglich (vgl Kapitel Datenflussanalyse). Worst-Case Laufzeit quadratisch bis kubisch.

Zugrundeliegender Verband: Potenzmengenverband der Grundblöcke (oder Anweisungsnummern $x_1, x_2, \ldots, x_n$)
Mit $z_i = dom(x_i)$ ist

$$f_i(z_1, z_2, \ldots, z_n) = \{x_i\} \cup \bigcap_{x_j \in \mathrm{pred}(x_i)} z_j$$

wobei f_i nicht von allen z_j abhängt, sondern nur von Vorgängern von x_i
f_i sind monoton (wieso?), $F = (f_1, f_2, \ldots, f_n)$ hat deshalb Fixpunkt

Der Lengauer-Tarjan-Algorithmus

Lengauer, T. und Tarjan, R. E. *A Fast Algorithm for Finding Dominators in a Flowgraph* (1979)

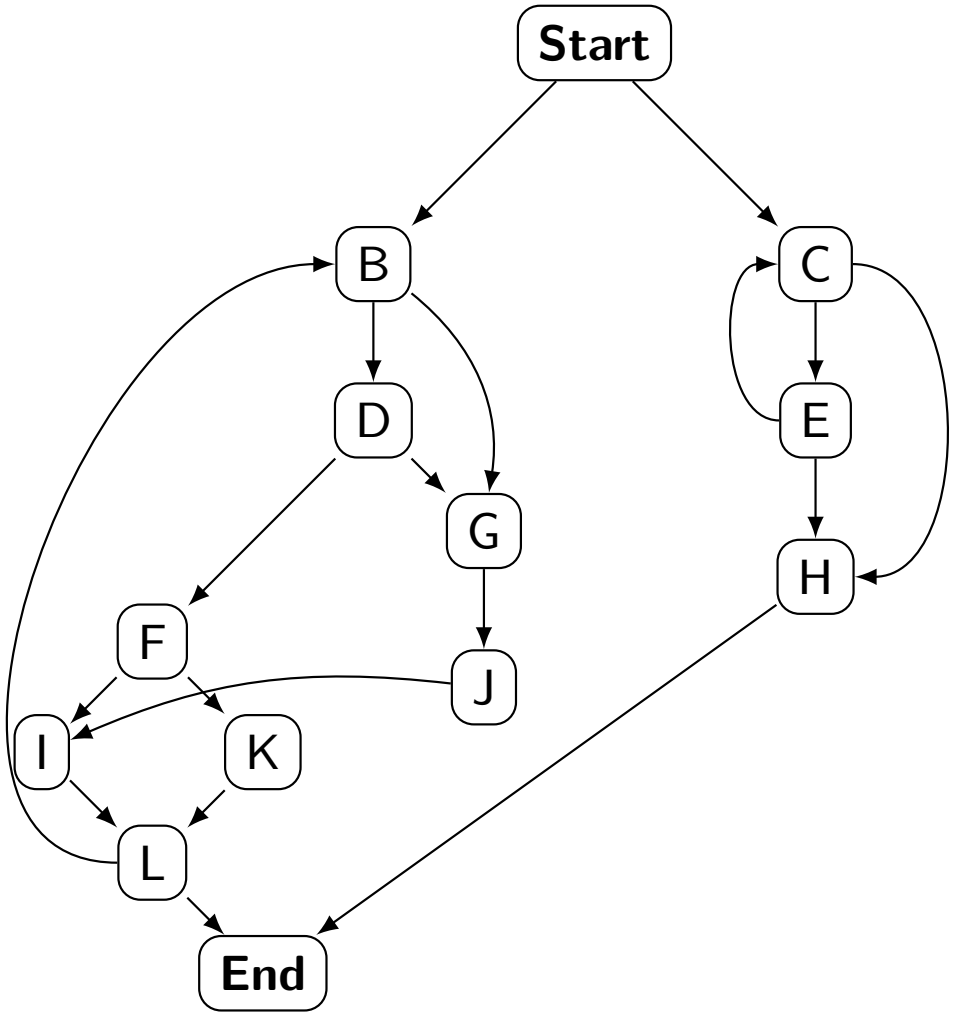
Tiefensuchbaum

Nimmt man die bei einer Tiefensuche besuchten Kanten und Knoten, so entsteht ein Tiefensuchwald. Auf Ablaufgraphen ein *Tiefensuchbaum*.

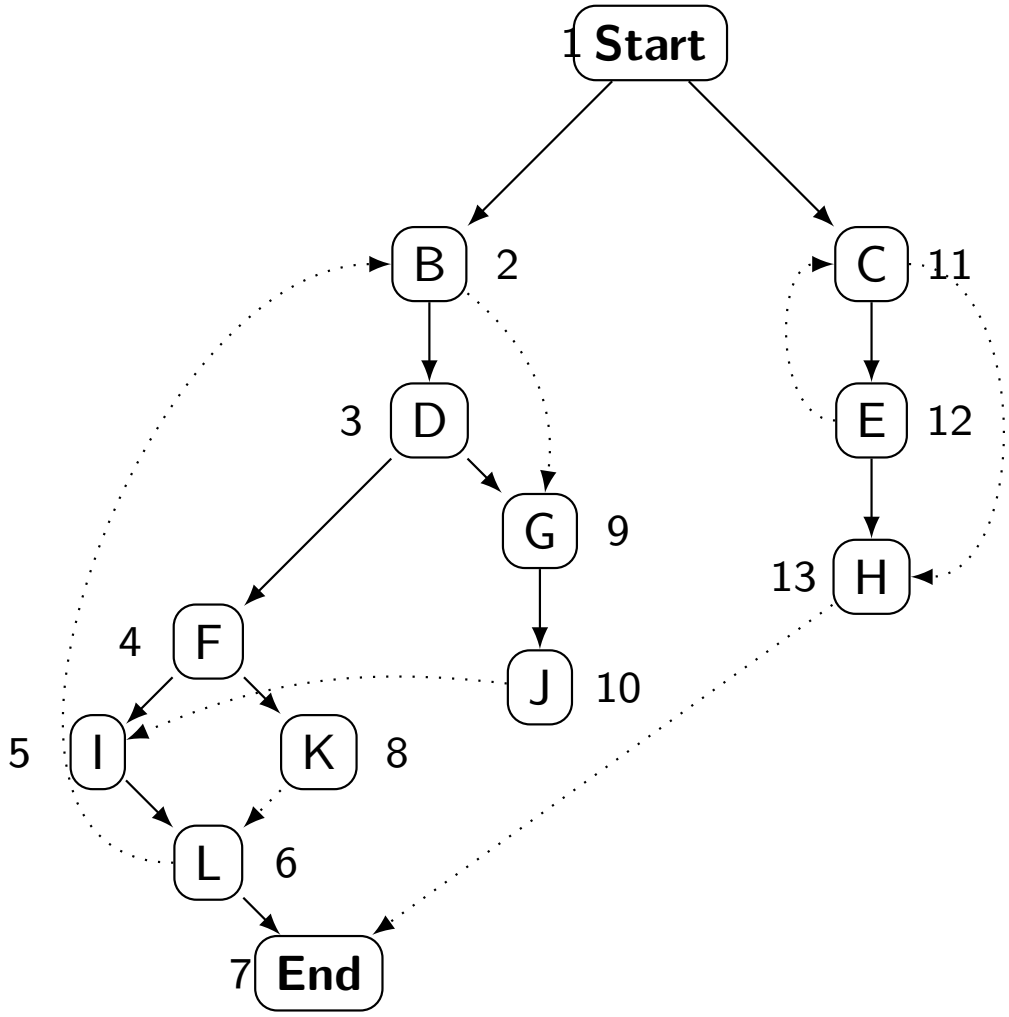
Tiefensuchnummern

Nummerieren der Knoten in Besuchsreihenfolge (pre-order Numbering) legt die *Tiefensuchnummern* der Knoten fest. Schreibweise: dfnum(x)

Beispiel - Ablaufgraph



Beispiel - Tiefensuchbaum mit Tiefensuchnummern



Tiefensuchbäume - Eigenschaften

Betrachte Knoten $n \neq$ **Start** und seinen Pfad im Spannbaum:

- Für alle Knoten k auf dem Pfad gilt: $dfnum(k) \leq dfnum(n)$.
- Alle Dominatoren von n liegen auf dem Pfad (folgt aus Definition des Dominators). Damit gilt: $d \preceq n \Rightarrow dfnum(d) \leq dfnum(n)$

$\Rightarrow$ Berechnung des direkten Dominators von n durch Testen der Vorgänger auf dem Pfad.

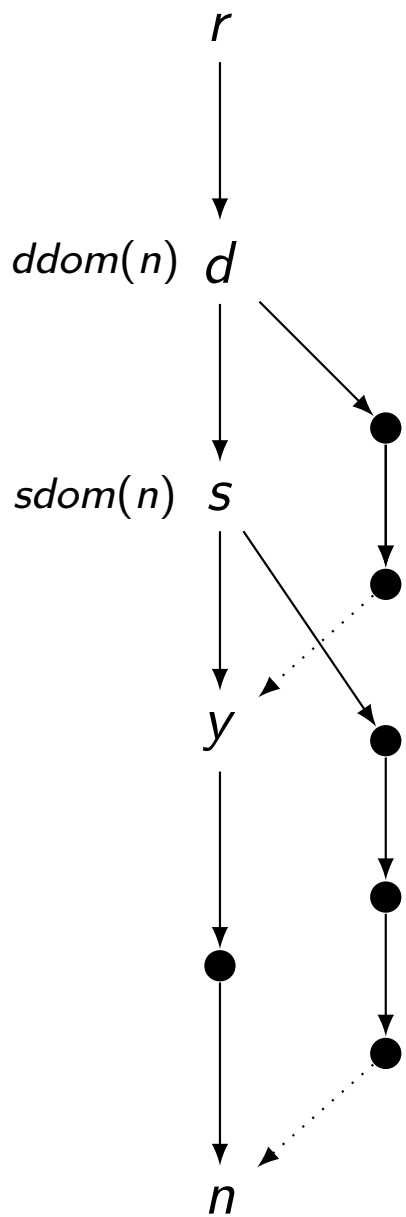
- Ein Knoten x des Pfades ist genau dann kein Dominator von n, wenn es alternative Pfade zu n um x herum gibt.
- Sei Pfad $a \rightarrow^* b$ vorhanden. Entweder ist $dfnum(a) < dfnum(b)$, dann liegt a im Tiefensuchbaum oberhalb b (Schreibweise: a anc b). Oder es ist $dfnum(a) > dfnum(b)$, dann wurde a später besucht und liegt „rechts neben“ b
- weiss man also, dass Pfad $a \rightarrow^* b$ vorhanden ist, so kann man mit dfnum-Vergleich einfach testen, ob a anc b

Semidominatoren

Beachte: Im folgenden identifizieren wir Knoten mit ihren Tiefensuchnummern!

Ein Semidominator ist definiert als

$$sdom(n) = min\{m | P = m, v_1, \ldots, v_k, n \text{ ist Pfad mit } v_i > n \text{ für } 1 \leq i \leq k\}$$



- $sdom(n)$ ist kleinster (i.e. oberster) Vorgänger von n auf dem Tiefensuchpfad, von dem Pfad $s \rightarrow^* n$ außerhalb des Tiefensuchpfades („Umleitung“) existiert.
- Der Semidominator muss nicht der direkte Dominator sein. (Beispiel siehe Bild).
- aber unterhalb $sdom(n)$ kann $ddom(n)$ nicht liegen!

Berechnung des Semidominators von n

$sdom(n)$ wird aus Semidominatoren der Vorgänger von n berechnet. Betrachte dazu jeden Vorgänger v von n.
$sdom(n)$ ist die „oberste Umleitung“ nach n über ein v. Für jedes v wird Menge von Kandidaten bestimmt, am Ende alle Kandidatenmengen vereinigt und darin $sdom(n)$ bestimmt

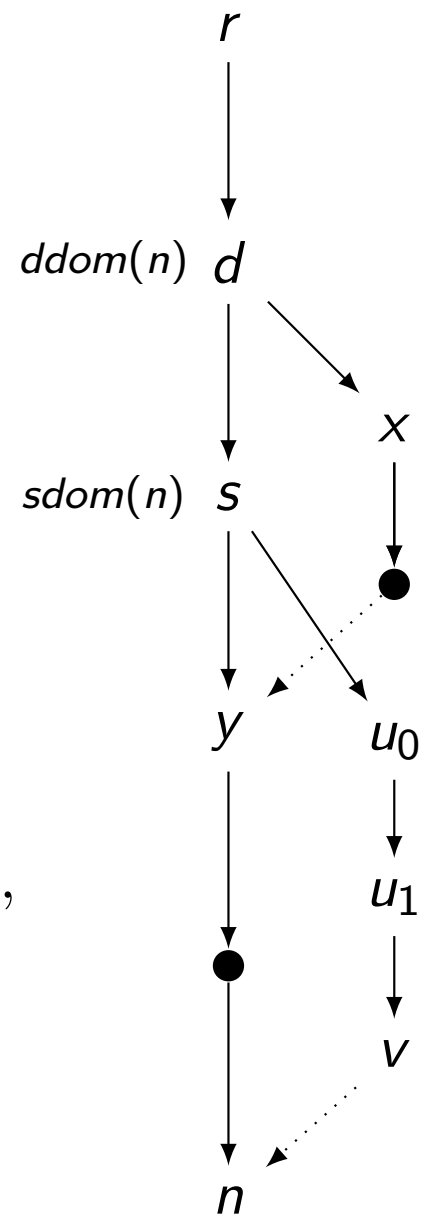
Fall 1:
Liegt v auf dem Tiefensuchpfad nach n (also $\text{dfnum}(v) < \text{dfnum}(n)$) so ist v ein Kandidat (denn v ist evtl der einzige Vorgänger)

Berechnung des Semidominators von $n/2$

Fall 2:
Liegt v nicht auf dem Tiefensuchpfad nach n (also $\text{dfnum}(v) > \text{dfnum}(n)$) so betrachte alle u auf dem Tiefensuchpfad nach v (incl v); für diese gilt $\text{dfnum}(u) \leq \text{dfnum}(v)$.
Alle $sdom(u)$ sind Kandidaten, denn es gibt dann Umleitung $sdom(u) \rightarrow^* u$, also auch Umleitung $sdom(u) \rightarrow^* u \rightarrow^* v \rightarrow n$, dh unterhalb dieser Kandidaten kann $dom(n)$ nicht liegen. Deshalb ist

$$sdom(n) = min\{sdom(u) \mid dfnum(u) \leq dfnum(v), v \rightarrow n \in CFG\}$$

$sdom(n)$ ist der Kandidat mit der kleinsten Tiefensuchnummer.



Implementierung Semidominatorberechnung

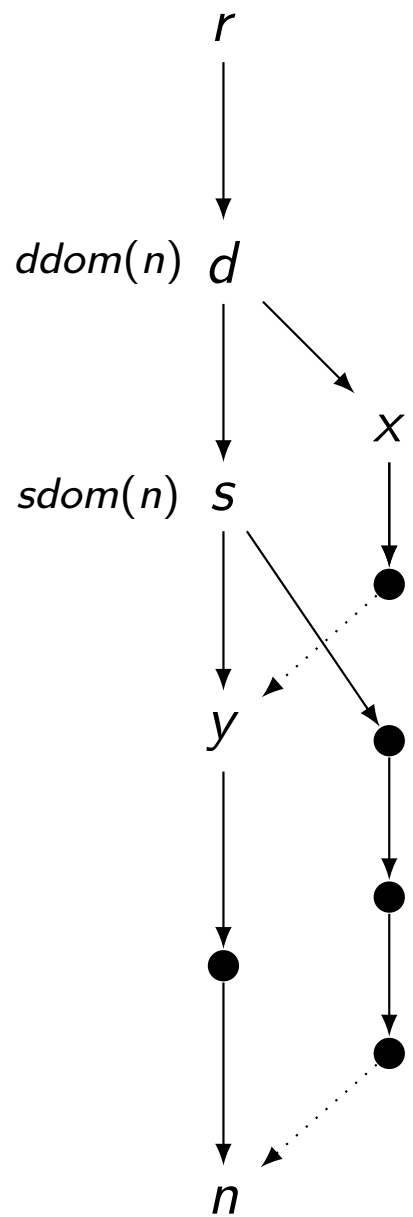
Implementierung: top-down Traversierung des Tiefensuchbaums; wenn man zu n kommt, sind die $sdom(u)$ bereits berechnet; man muss nur noch die Ketten $n \longrightarrow v \longrightarrow^* u \ldots$ rückwärts laufen und dabei Minimum der $dfnum(sdom(u))$ bestimmen

Dominator Theorem

Auf dem Tiefensuchpfad zwischen $sdom(n)$ und n einschließlich, sei y derjenige Knoten dessen Semidominator die kleinste Tiefensuchnummer besitzt. Dann ist

$$ddom(n) = \begin{cases} sdom(n) & \text{if } sdom(y) = sdom(n) \\ ddom(y) & \text{if } sdom(y) \neq sdom(n) \end{cases}$$

Also ist $ddom(n) = sdom(y)$ mit $y \in sdom(n) \rightarrow^* n$ und $sdom(y)$ minimal (i.e. möglichst weit oben)
Beweis: LT79



Dominator Theorem/2

Zur Erläuterung:

1. Wenn $sdom(y) = sdom(n)$, so gibt es keinen Umweg $x \rightarrow^* y$ mit x oberhalb $sdom(n)$. Deshalb auch keinen $x \rightarrow^* z$ mit z unterhalb y auf dem Tiefensuchpfad, denn y hat minimalen Semidominator. Also auch keinen $x \rightarrow^* z \rightarrow^* n$. Deshalb ist $sdom(n) = ddom(n)$.
2. Sonst gilt: $ddom(n)$ kann nicht unterhalb $ddom(y)$ auf Tiefensuchpfad liegen, da es ja Umweg $ddom(y) \rightarrow^* y \rightarrow^* n$ gibt. $ddom(n)$ kann nicht oberhalb liegen, da y oberhalb n liegt und dehalb muss der direkte (!) Dominator von y oberhalb des direkten Dominators von n liegen

Implementierung: ähnlich wie Semidominatoren. Das wiederholte Ablaufen der Kette $n \rightarrow y \rightarrow \ldots$ kann Faktor $O(n)$ kosten
$\Longrightarrow$ Pfadkompression (s.u.)

Berechnung der direkten Dominatoren

- Tiefensuche, dfnums
- Konstruktion des Dominanzbaums bottom-up als aufspannender Baum in inverser dfnum-Reihenfolge
- dabei Berechnung der Semidominatoren gemäß Formel
- Pfadkompression (ähnlich wie bei Union-Find) zur Berechnung der Pfade $sdom(n) \rightarrow^* n$ anwenden.
- Komplexität: $O(n \cdot \ln n)$, mit balancierter Pfadkompression: $O(n \cdot \alpha(n))$ ($\alpha =$ inverse Ackermann-Funktion; für alle praktischen Fälle ≤ 7)