Anforderungen an Zwischendarstellungen

widersprüchliche Anforderungen:

1. möglichst quellennah, da Informationen an Wartungsprogrammierer ausgegeben werden sollen bzw. tatsächlicher Code wieder generiert werden soll

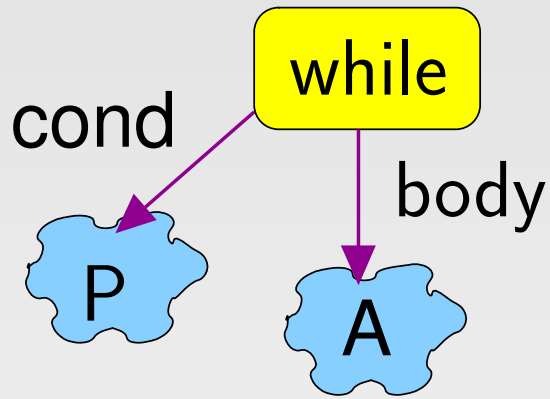
$\Rightarrow$ alle spezifischen Konstrukte müssen dargestellt werden

2. möglichst vereinheitlicht, um das Schreiben von Programmanalysen für verschiedene Programmiersprachen zu vereinfachen

$\Rightarrow$ möglichst wenige, allgemeine Konstrukte

Quellennahe Betrachtung

while P **loop**
  A;
end loop;

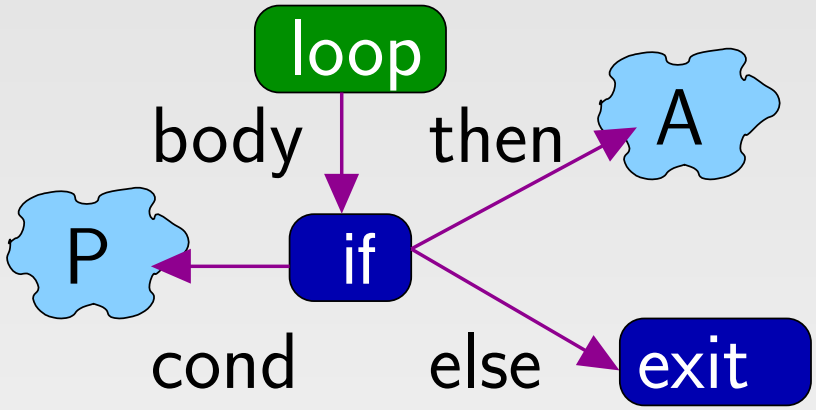


Einfache allgemeine Konstrukte

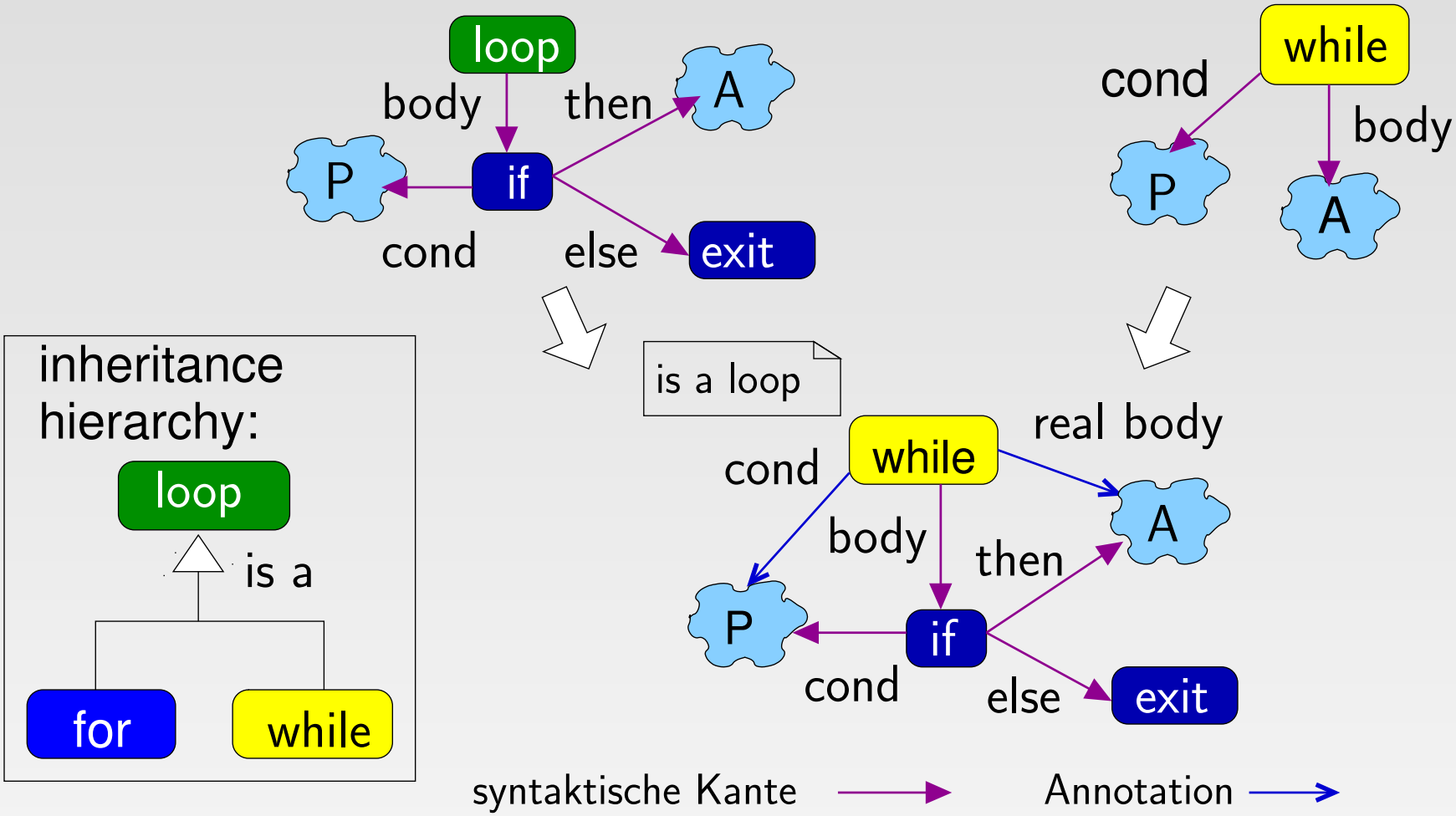
while P **loop**
  A;
end loop;

ist äquivalent zu:

loop
  if P **then**
    A;
  else
    exit;
  end if;
end loop;



Vereinigung der Sichten



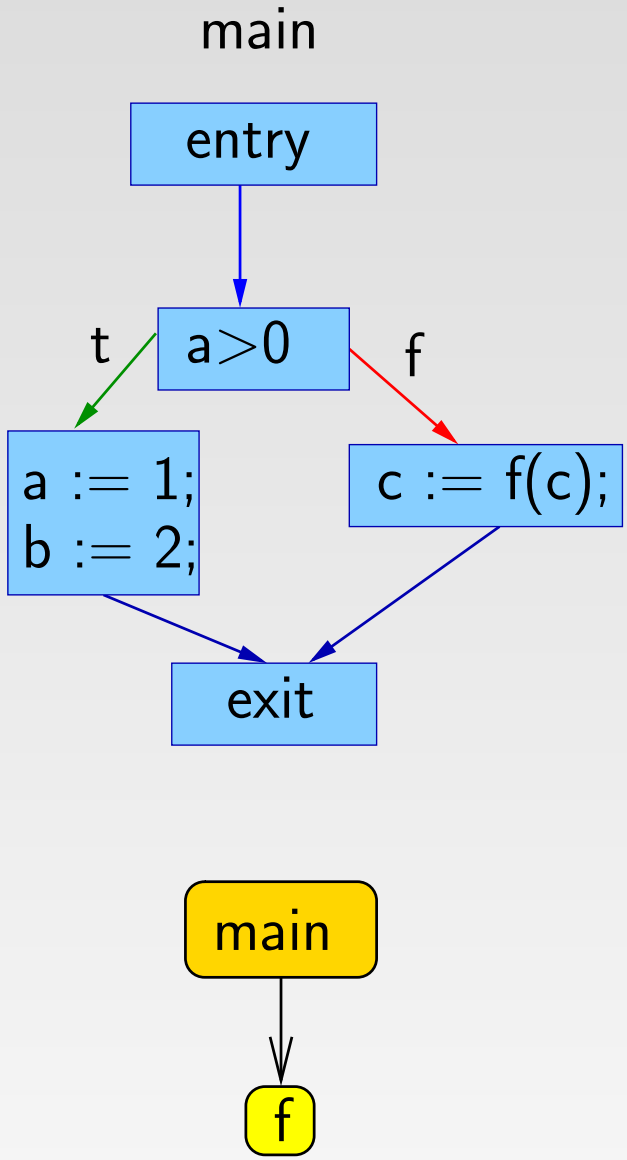
Zwischendarstellungen

Im Folgenden präsentieren wir die wichtigsten Elemente der Darstellung dynamischer Semantik und deren Herleitung ...

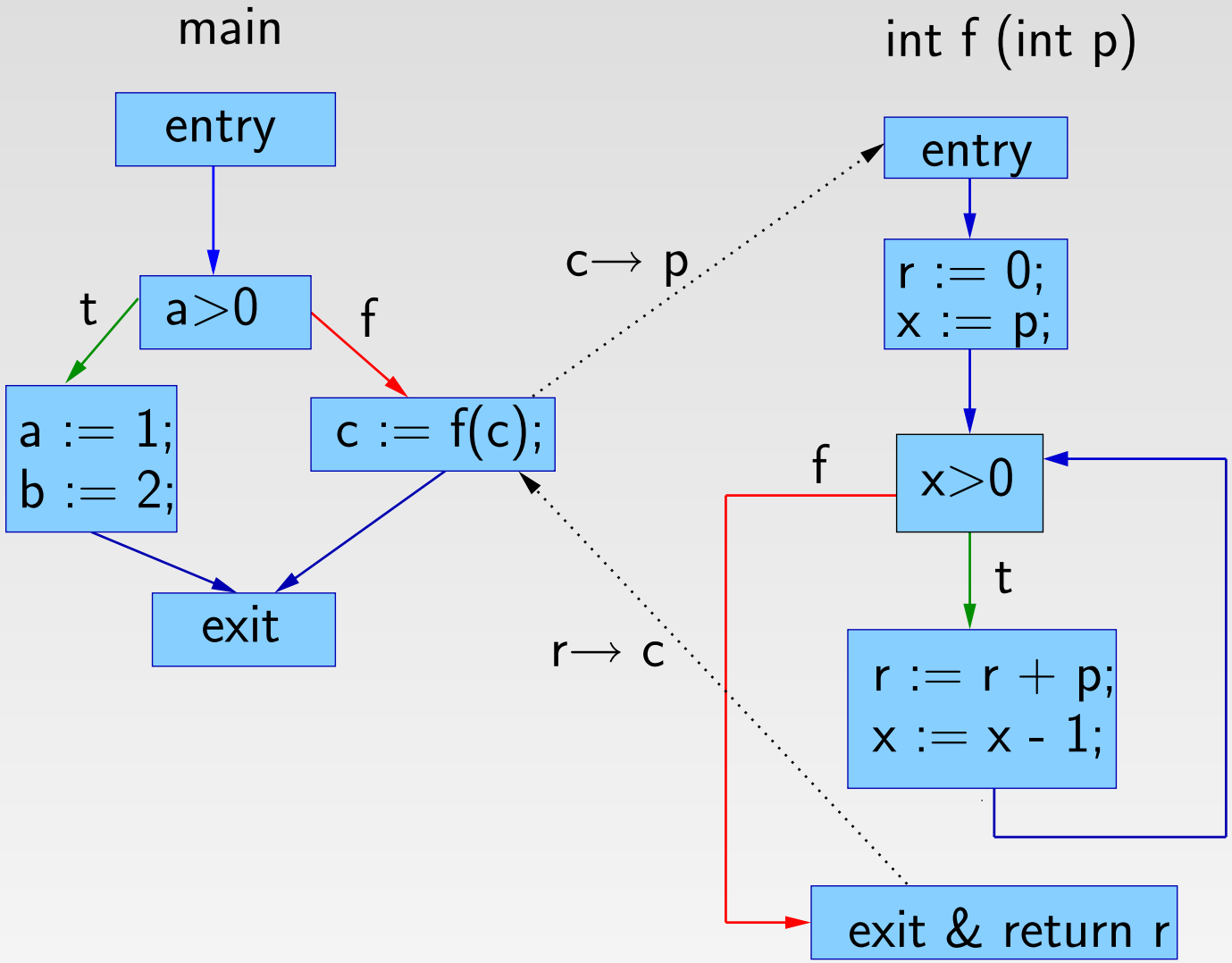
1. Kontrollflussanalysen
2. Datenflussanalysen

Kontrollflussinformation

- intraprozedural: Flussgraph
  - Knoten: Grundblöcke
  - Kanten: (bedingter/unbedingter) Kontrollfluss
  - ergibt sich aus syntaktischer Struktur und etwaigen Gotos, Exits, Continues, etc.
- interprozedural: Aufrufgraph
  - Multigraph
  - Knoten: Prozeduren
  - Kanten: Aufruf (eine für jede Aufrufstelle); Schleifen $\rightarrow$ Rekursion
  - ergibt sich aus expliziten Aufrufen im Programmcode sowie Aufrufe über Funktionszeiger



Intra- und interprozeduraler Kontrollfluss



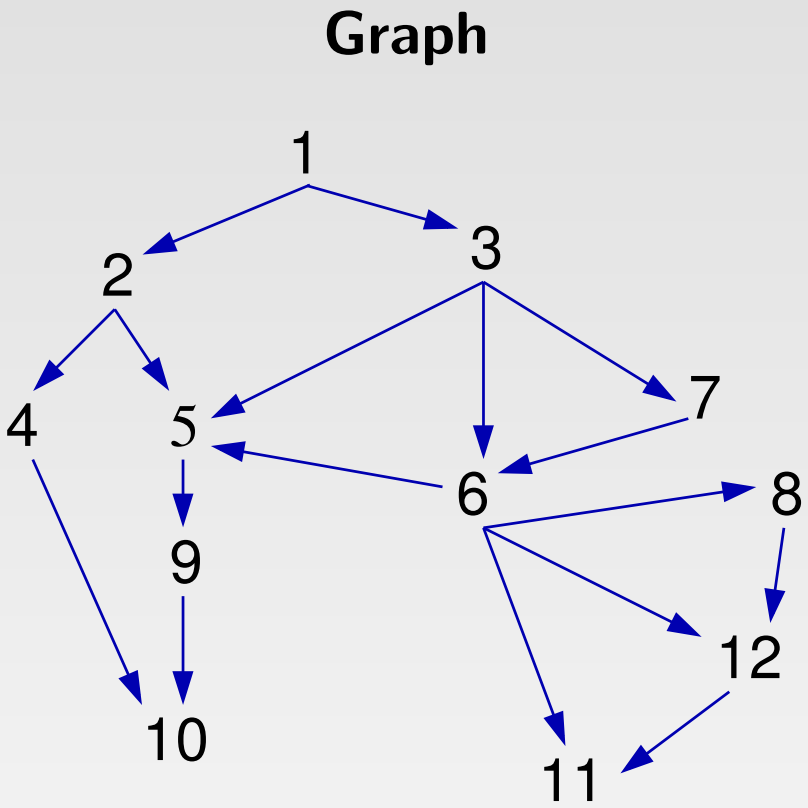
Dominanz

- Fragestellungen:
  - Welche Prozeduren sind lokal zueinander?
    Genauer: Gibt es eine Prozedur D, über nur die ein Aufruf von N erfolgt?
  - Welcher Block D im Flussgraph muss in jedem Falle passiert werden, damit Block N ausgeführt werden kann?

Dominanz

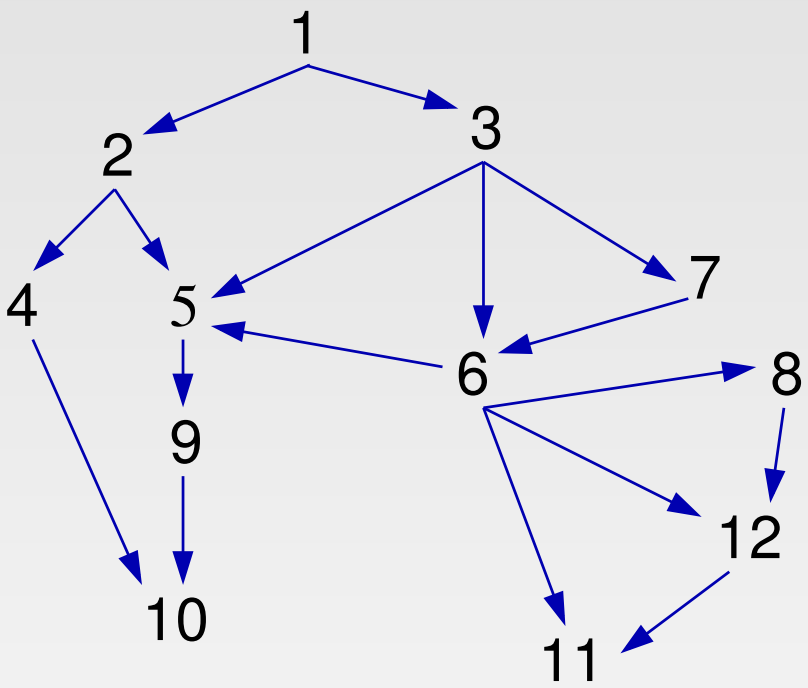
- Fragestellungen:
  - Welche Prozeduren sind lokal zueinander?
    Genauer: Gibt es eine Prozedur D, über nur die ein Aufruf von N erfolgt?
  - Welcher Block D im Flussgraph muss in jedem Falle passiert werden, damit Block N ausgeführt werden kann?
- Antwort: D ist der Dominator von N
  - Ein Knoten D **dominiert** einen Knoten N, wenn D auf allen Pfaden vom Startknoten zu N liegt.
  - Ein Knoten D ist der **direkte Dominator** von N, wenn
    1. D dominiert N und
    2. alle weiteren Dominatoren von N dominieren D

Dominanz

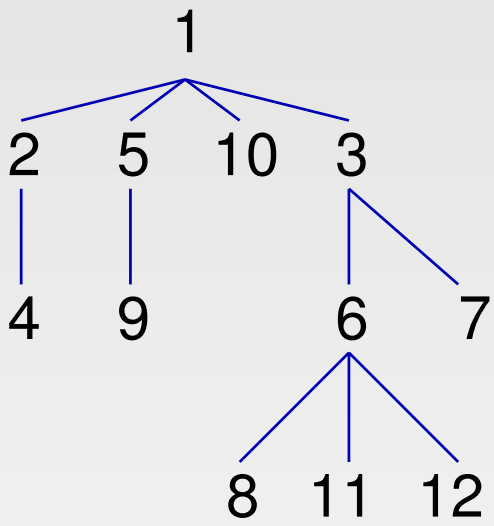


Dominanz

Graph

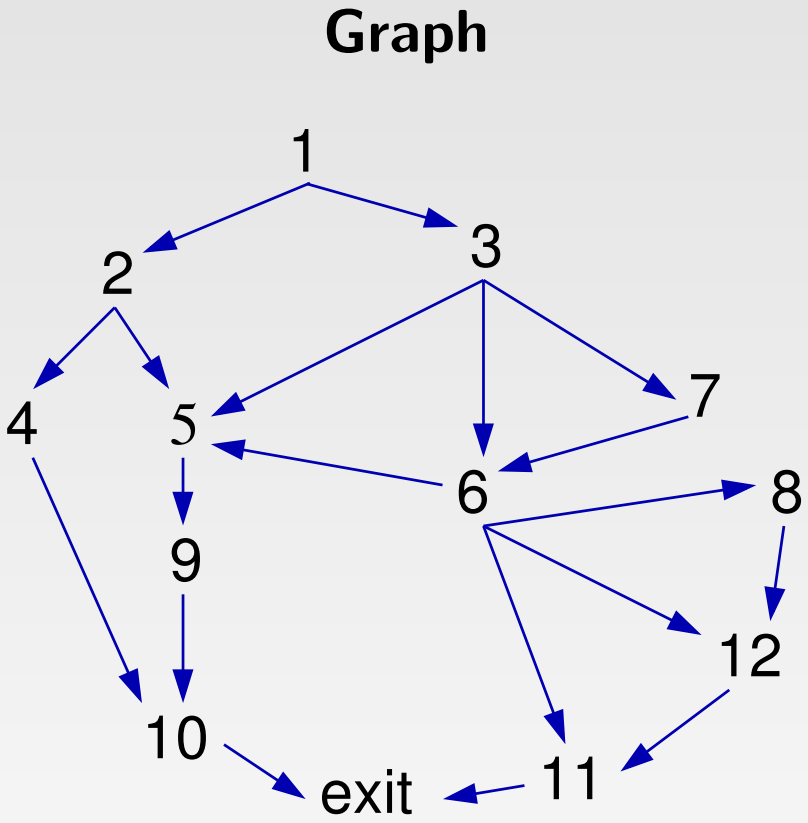


Dominanzbaum



Postdominanz

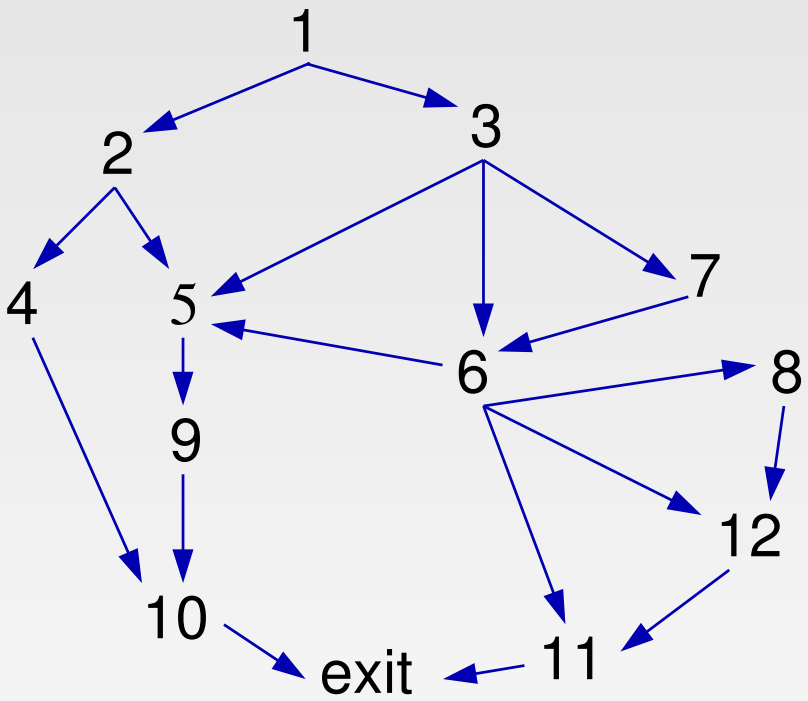
Ein Knoten D **postdominiert** einen Knoten N, wenn jeder Pfad von N zum Endknoten den Knoten D enthält (entspricht Dominanz des umgekehrten Graphen).



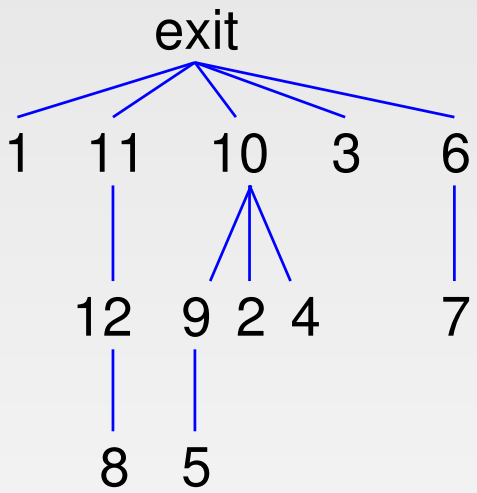
Postdominanz

Ein Knoten D **postdominiert** einen Knoten N, wenn jeder Pfad von N zum Endknoten den Knoten D enthält (entspricht Dominanz des umgekehrten Graphen).

Graph



Postdominanzbaum



Kontrollabhängigkeit I

Kontrollabhängigkeit = Bedingung B, von welcher die Ausführung eines anderen Knotens X abhängt.

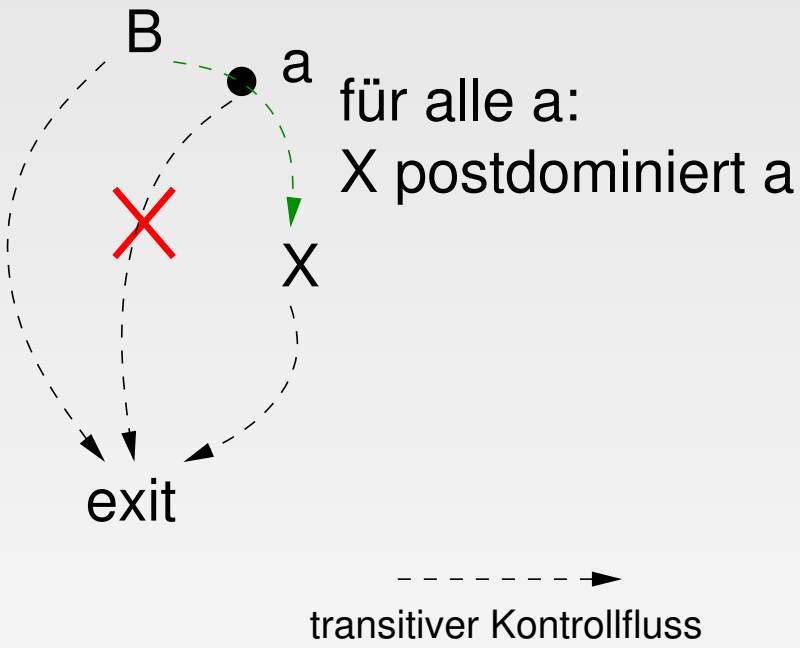
1. B muss mehrere direkte Nachfolger haben und
2. B hat einen Pfad zum Endknoten, der X vermeidet (d.h. B kann nicht von X postdominiert werden) und
3. B hat einen Pfad zu X, d.h. insgesamt hat B mindestens 2 Pfade:
   - einer führt zu X
   - einer umgeht X

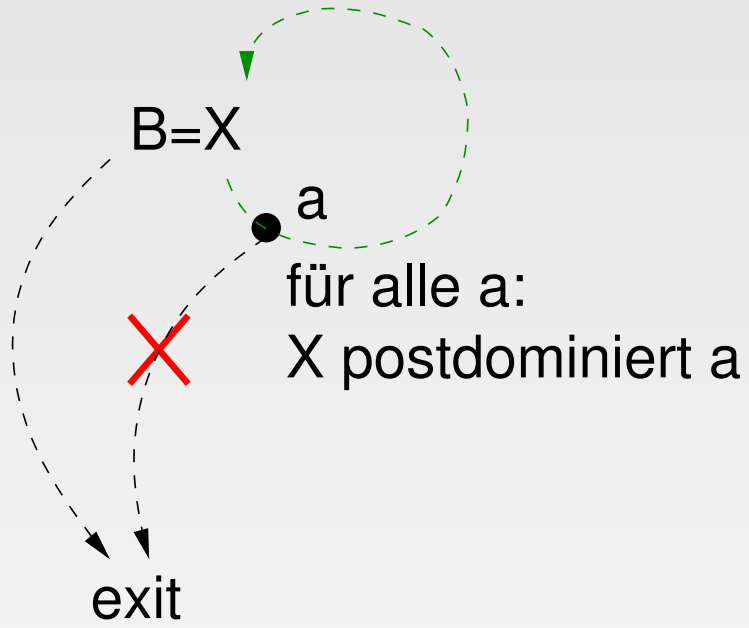
   und
4. B ist der letzte Knoten mit einer solchen Eigenschaft

Kontrollabhängigkeit II

Ein Knoten X ist direkt kontrollabhängig von einem Knoten B genau dann, wenn

1. es einen nicht-leeren Pfad von B nach X gibt, so dass X jeden Knoten auf dem Pfad (ohne B) postdominiert und
2. entweder X = B oder X postdominiert B nicht

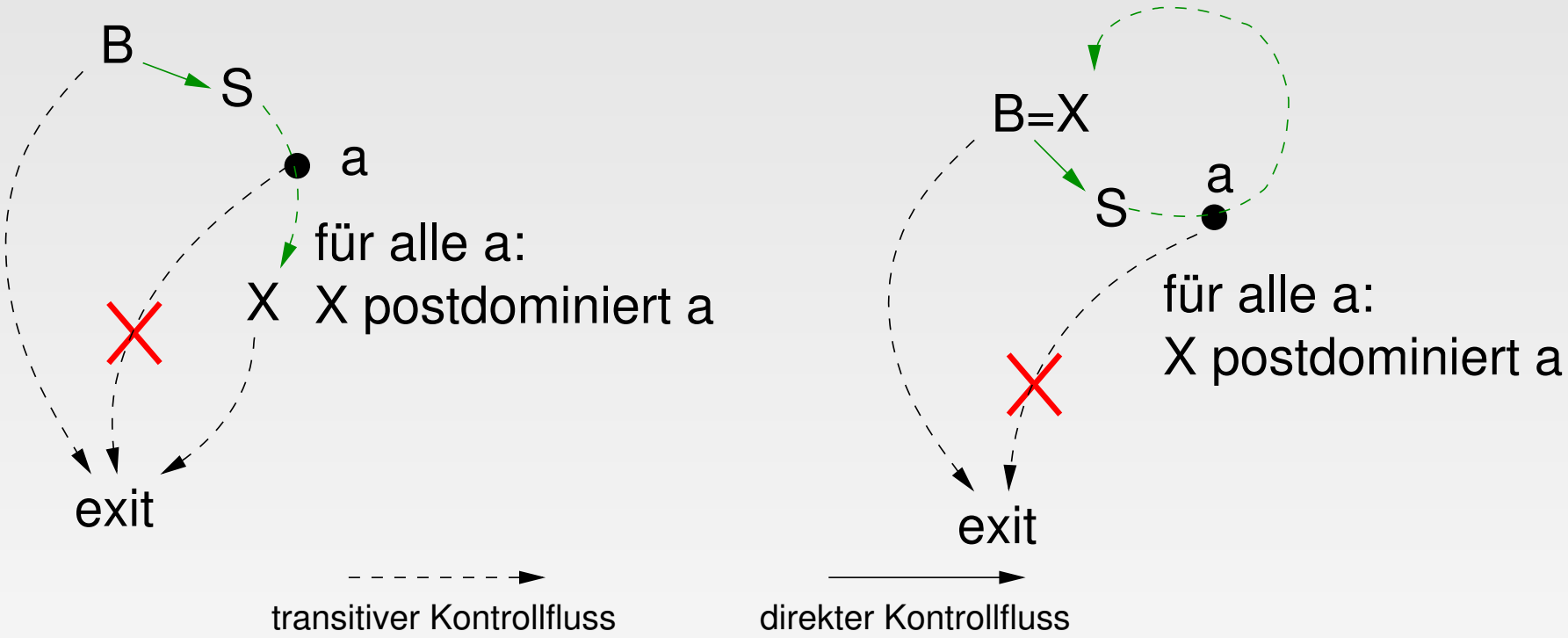




Kontrollabhängigkeit III

Ein Knoten X ist direkt kontrollabhängig von einer Kante (B, S) genau dann, wenn

1. es einen nicht-leeren Pfad beginnend mit (B, S) nach X gibt, so dass X jeden Knoten auf dem Pfad (ohne B) postdominiert und
2. entweder X = B oder X postdominiert B nicht



Kontrollabhängigkeit IV

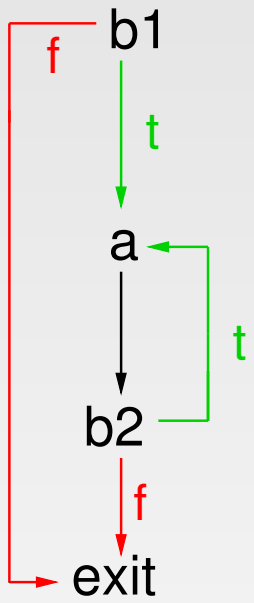
```
function CD ( (B, S) : Edge)
                    return list of nodes
  begin -- control dependency

   depends := empty_set;
   X := S;

   while X /= pdom (B) loop
     add X to depends;
     X := pdom (X);
  end loop;

  return depends;
end CD;
```

Kontrollabhängigkeit V

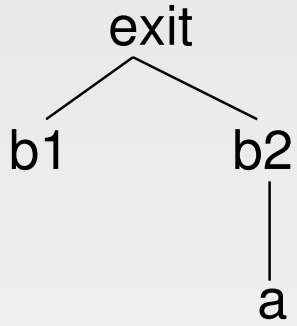
Beispiel für Kontrollabhängigkeit

```
if (b1) {
   do {
      a;
   } while (b2);
}
```

KFG



Postdominanz



(b1, a)	a, b2
(b1, exit)	–
(b2, a)	a, b2
(b2, exit)	–