Goto-Entfernen für den Comrade-Compiler

Nicolas Weber

Erinnerung CMDFG-Konstruktion

Erinnerung CMDFG-Konstruktion

- C → AST → CFG → SSA form → C(M)DFG

Erinnerung CMDFG-Konstruktion

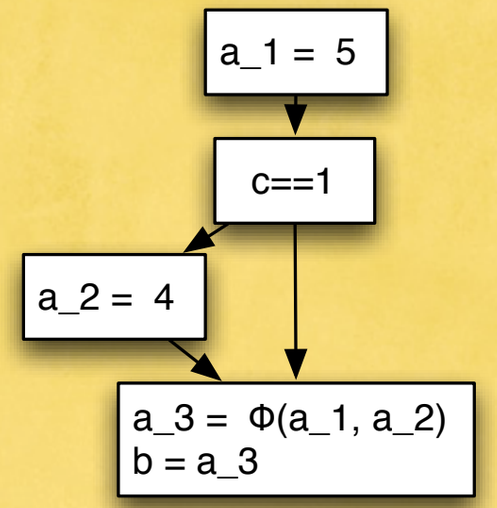
- C → AST → CFG → SSA form → C(M)DFG
- Dominance, Dominance Frontier

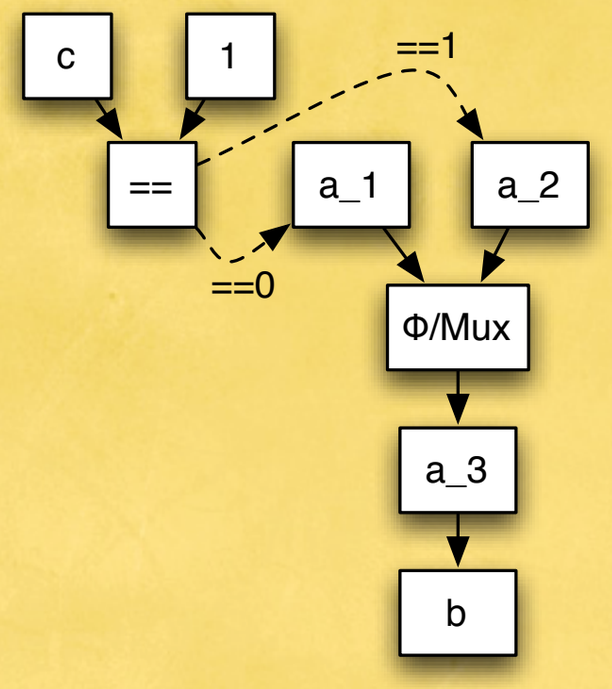
Erinnerung CMDFG-Konstruktion

- C → AST → CFG → SSA form → C(M)DFG
- Dominance, Dominance Frontier
- B control-dependent on A ⇔ A heißt Ferrante-Controller von B

Erinnerung CMDFG-Konstruktion

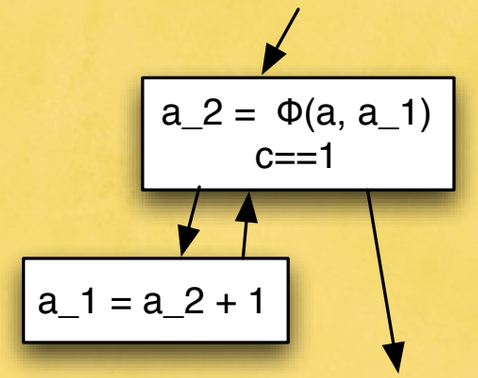
```
a = 5;
if (c == 1)
  a = 4;
b = a;
```

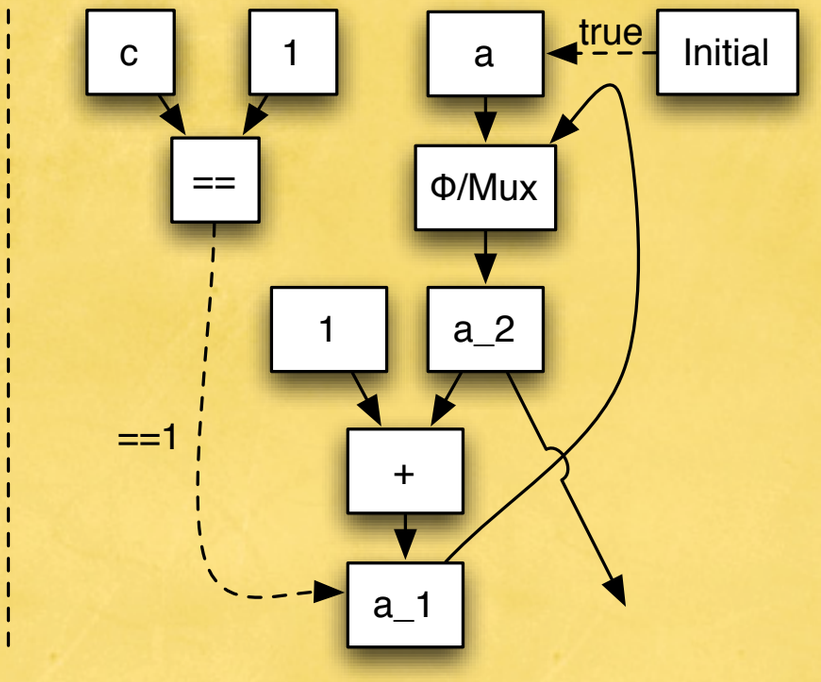




Erinnerung CMDFG-Konstruktion

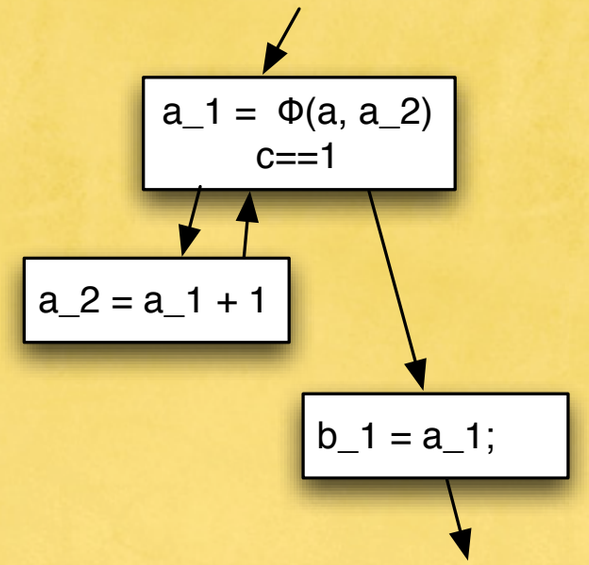
```
a = ...
while (c == 1)
  a = a + 1;
```

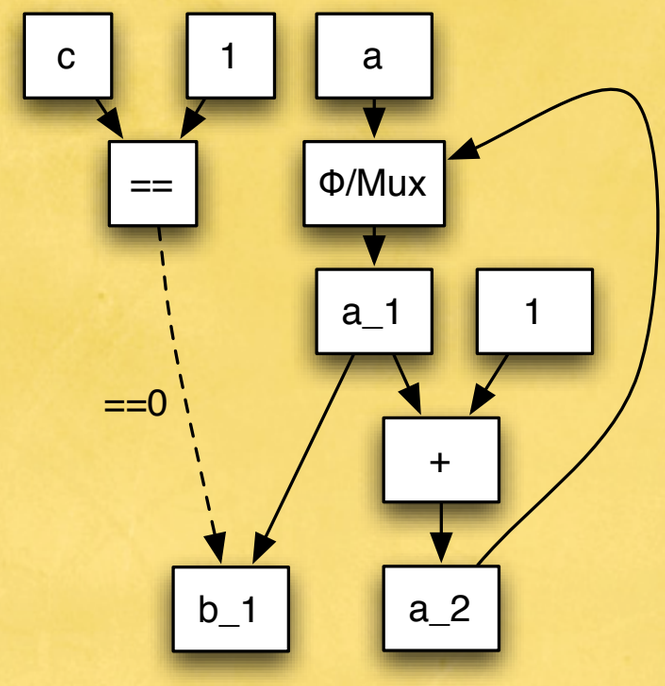




Erinnerung CMDFG-Konstruktion

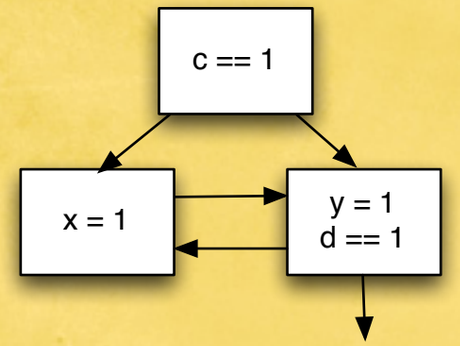
```
a = ...
while (c == 1)
  a = a + 1;
b = a;
```





Problem

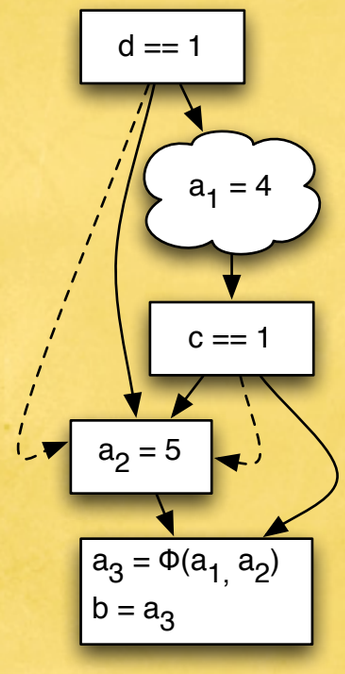
```
    if (c) goto b;
a:  x = 1;
b:  y = 1;
    if (d) goto a;
```



(CFG)

2

```
if (d == 1) goto L;
/* ... */
a = 4;
/* ... */
if (c == 1) {
L: a = 5;
}
b = a;
```



(CFG)

Problem

Problem

- Wir benötigen einen strukturierten CFG

Problem

- Wir benötigen einen strukturierten CFG
- Jede Schleife nur einen Eingang und einen Ausgang

Problem

- Wir benötigen einen strukturierten CFG
  - Jede Schleife nur einen Eingang und einen Ausgang
  - Jede Verzweigung genau eine Vereinigung

Lösungsidee: Control Flow Unification

- Gotos entfernen, Effekt über Flags simulieren
- Für jedes Label ein Flag
- Am Ende hat jede Schleife einen Ein-/Ausgang, zu jedem Join gehört genau ein If/Switch

Lösungsidee: Control Flow Unification

- Algorithmus von Erosa und Hendren 93
- Leicht modifiziert

AST-Basiert

- Baut sich besser in Comrade-Flow ein (Profiler)
- Source2source: Leicht zu verstehen, orthogonal zu bisherigem Ansatz
- Nicht nur Vorteile:

```
a: x = 4; goto b;
c: z = 5; goto d;
b: y = 6; goto c;
d:
```

Initialisierung

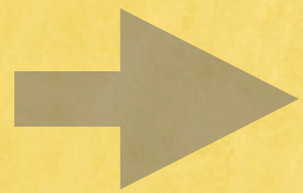
- Jedes Goto in ein `if (1)` einpacken (“ifgoto”)
- Breaks und Continues in gotos verwandeln (macht SUIF vollautomatisch)
- Breaks von Switch-Statements behalten

Einfache Fälle

```
if (c) goto forward;
 /* ... */
forward:
```

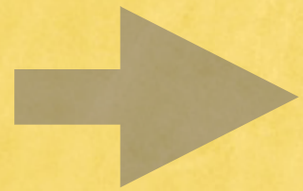
Einfache Fälle

```
 if (c) goto forward;
  /* ... */
forward:
```



Einfache Fälle

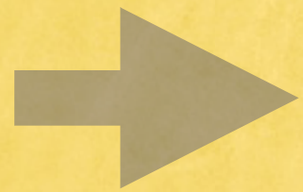
```
if (c) goto forward;
  /* ... */
forward:
```



```
if (!c) {
  /* ... */
}
```

Einfache Fälle

```
if (c) goto forward;
  /* ... */
forward:
```

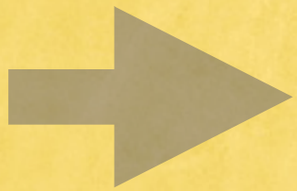


```
if (!c) {
  /* ... */
}
```

```
backward:
  /* ... */
  if (c) goto backward;
```

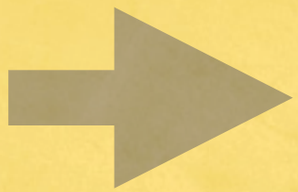
Einfache Fälle

```
if (c) goto forward;
  /* ... */
forward:
```



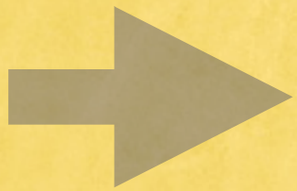
```
if (!c) {
  /* ... */
}
```

```
backward:
  /* ... */
  if (c) goto backward;
```



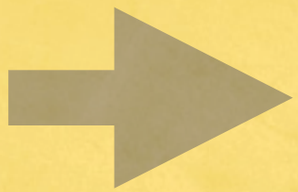
Einfache Fälle

```
if (c) goto forward;
  /* ... */
forward:
```



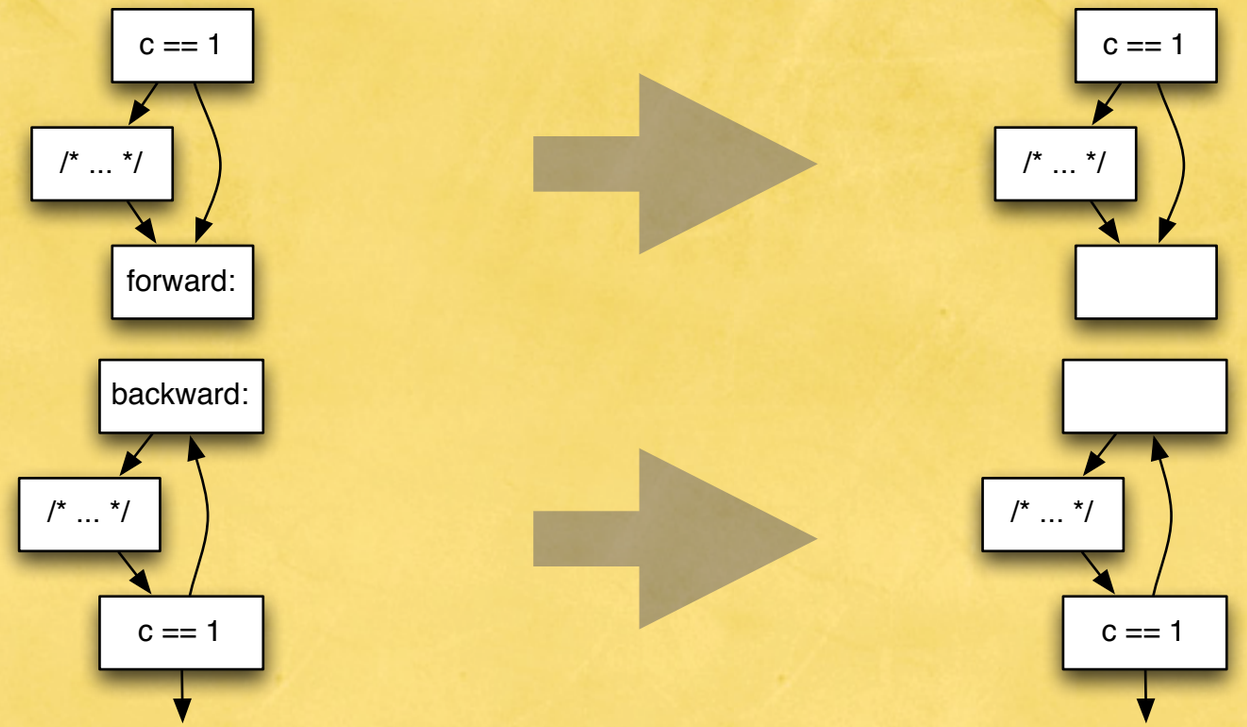
```
if (!c) {
  /* ... */
}
```

```
backward:
  /* ... */
  if (c) goto backward;
```

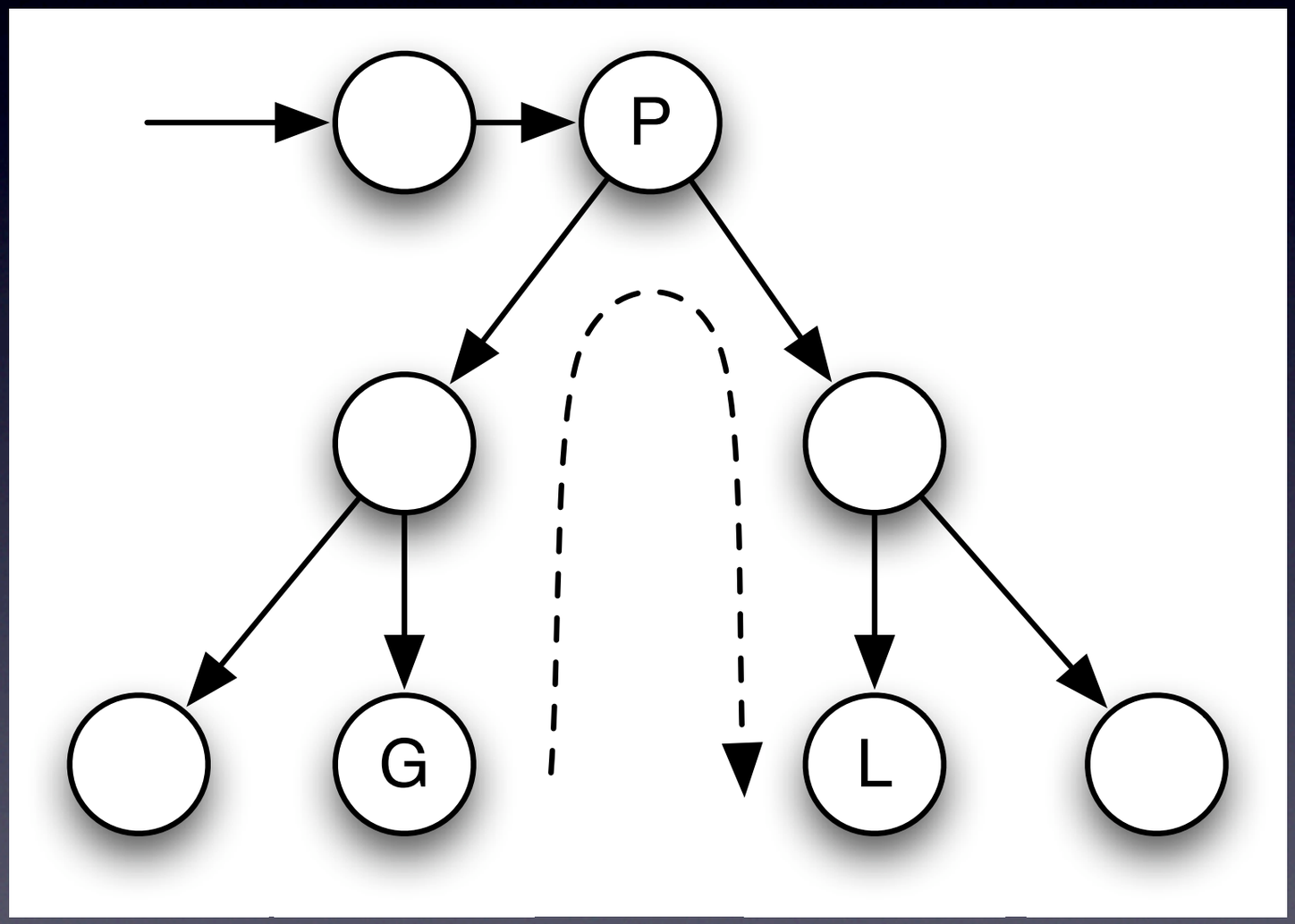


```
do {
  /* ... */
} while (c);
```

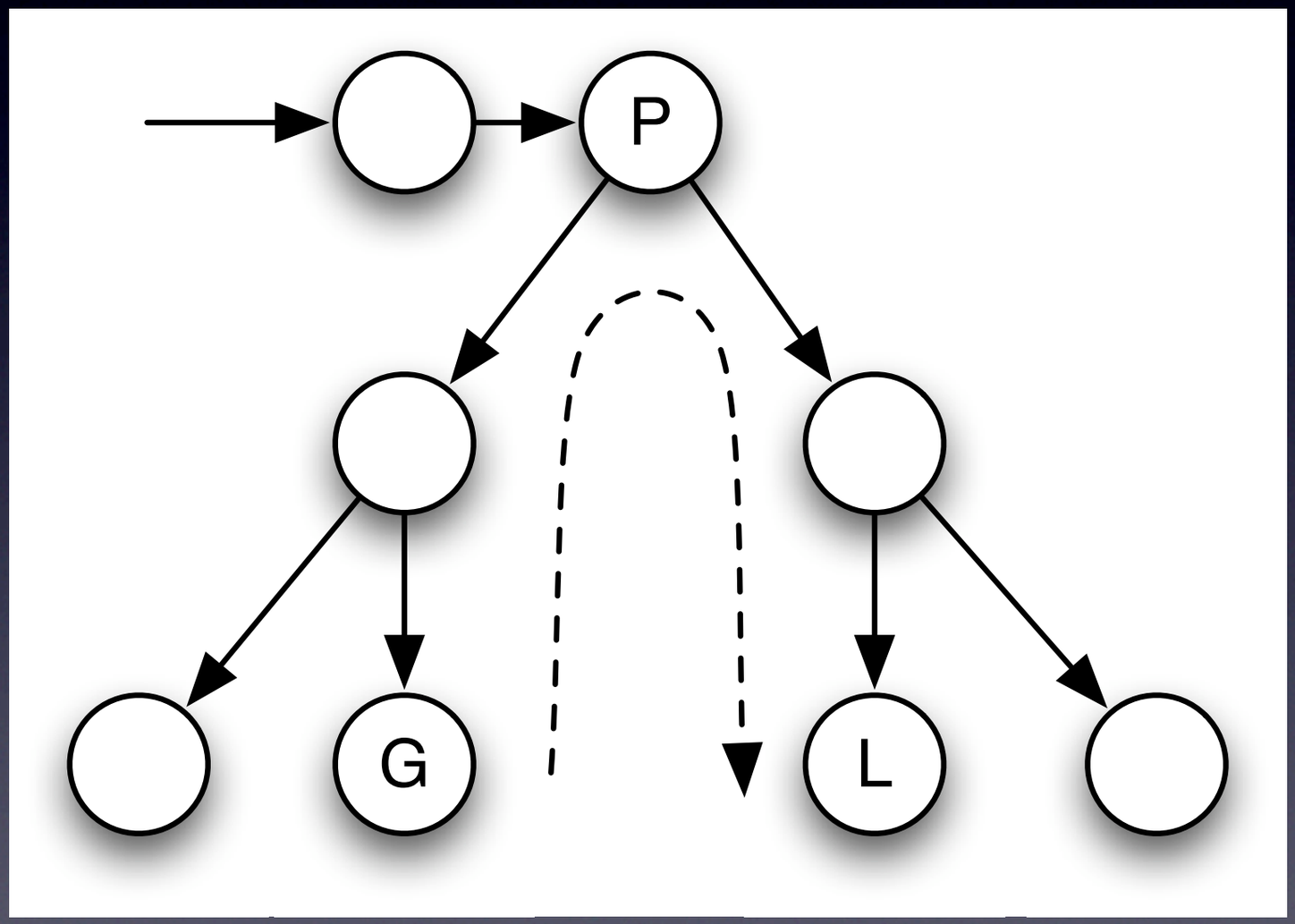
Einfache Fälle



Nicht alles ist einfach

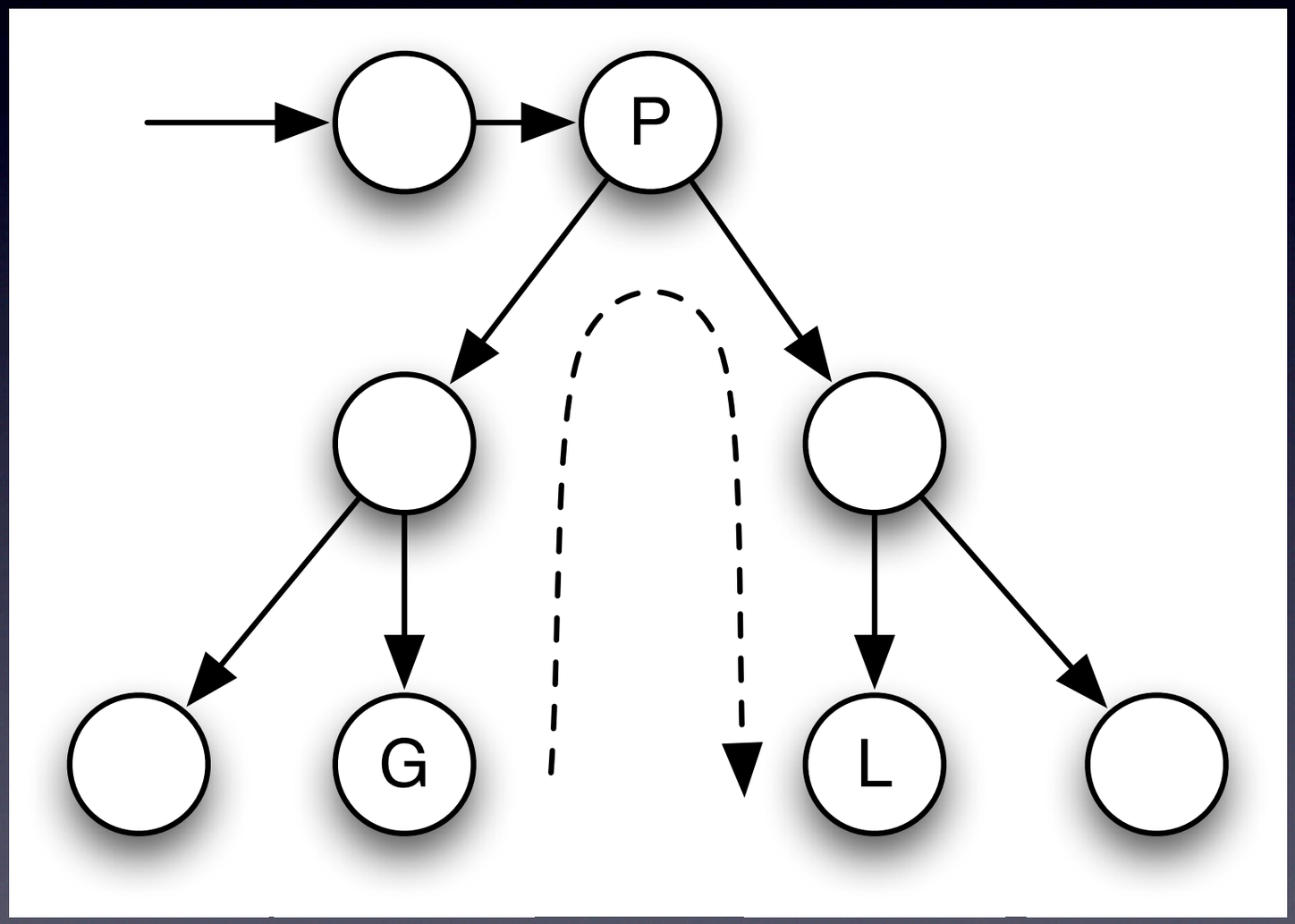


Nicht alles ist einfach



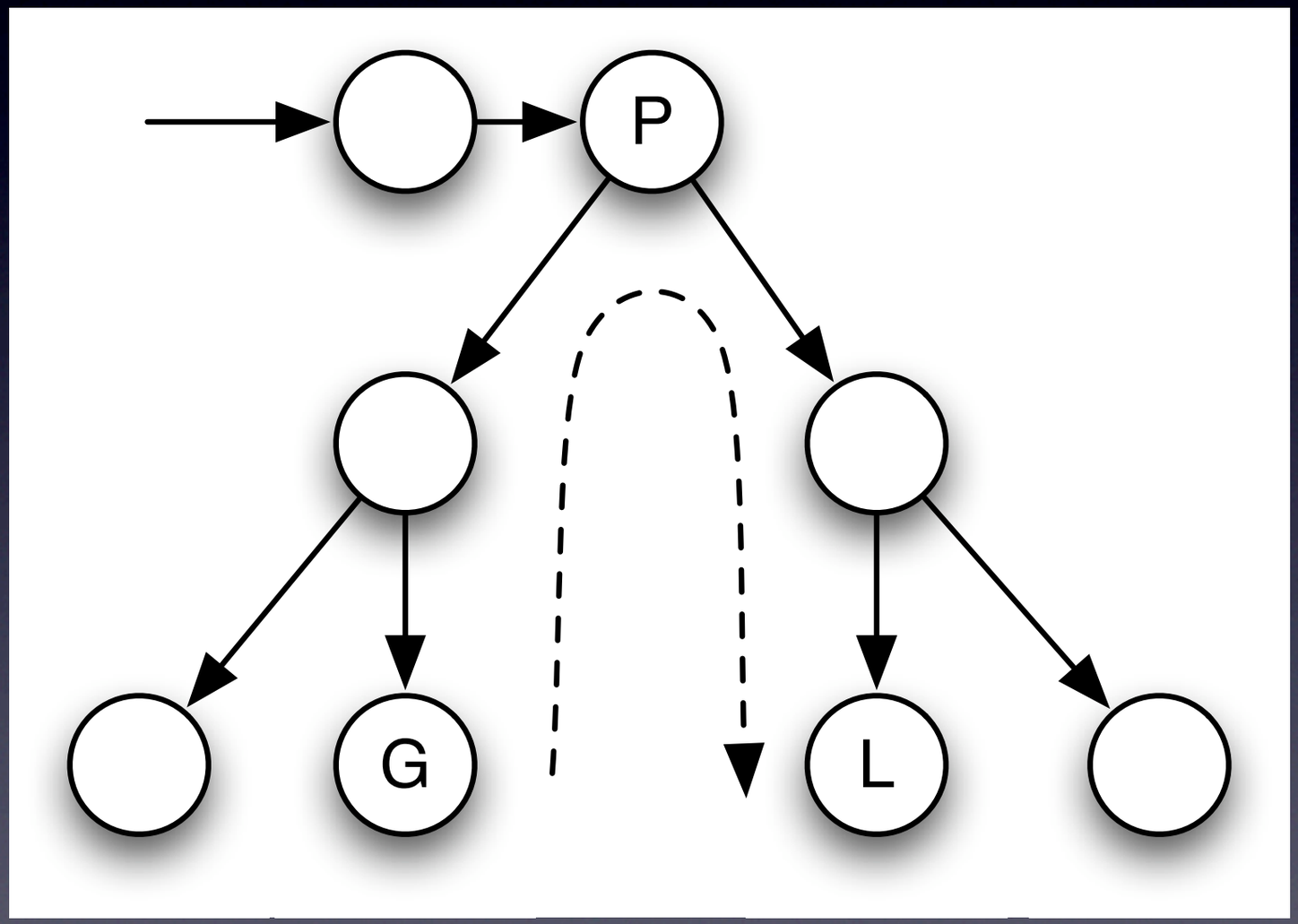
- Block

Nicht alles ist einfach



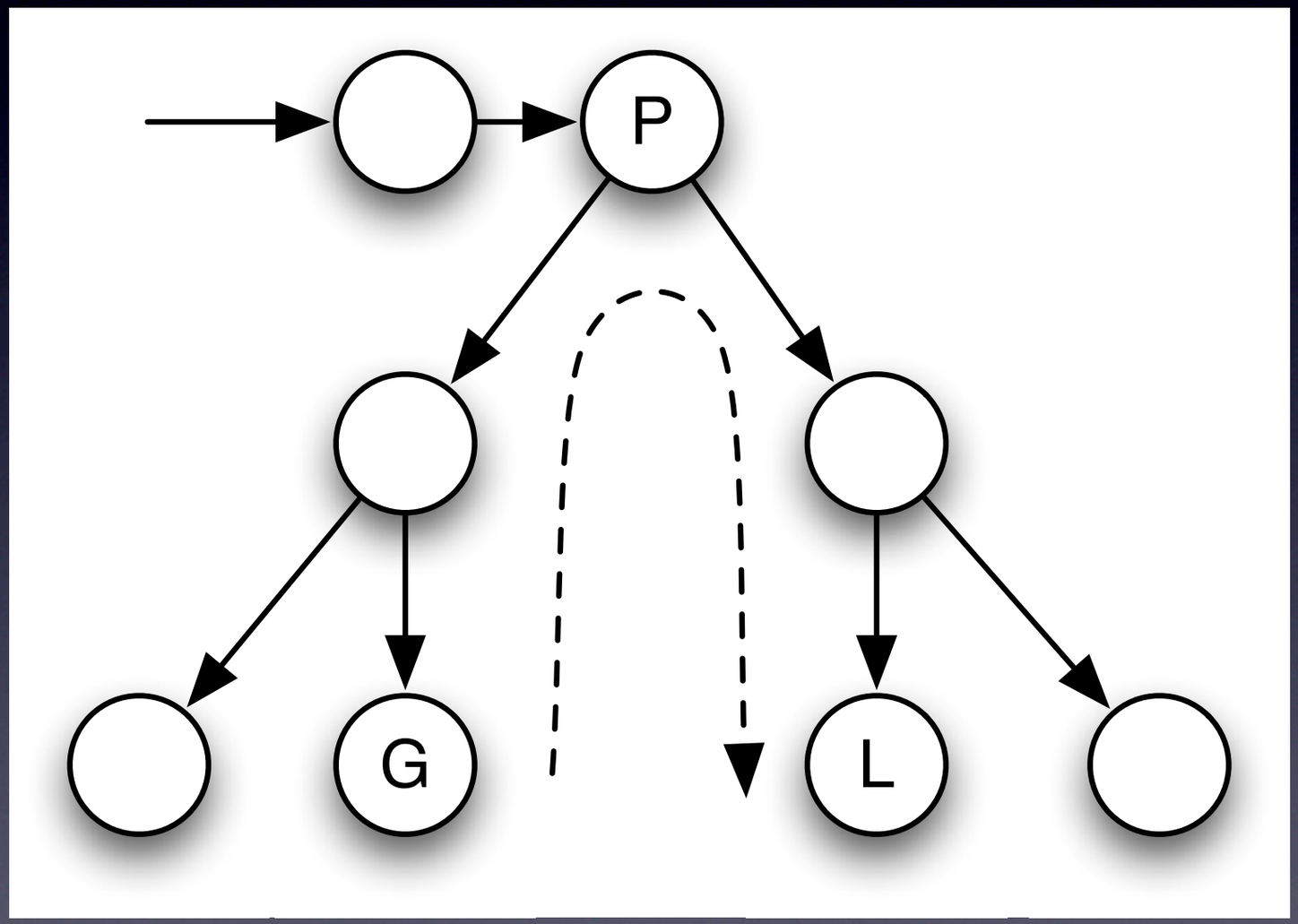
- Block
- If

Nicht alles ist einfach



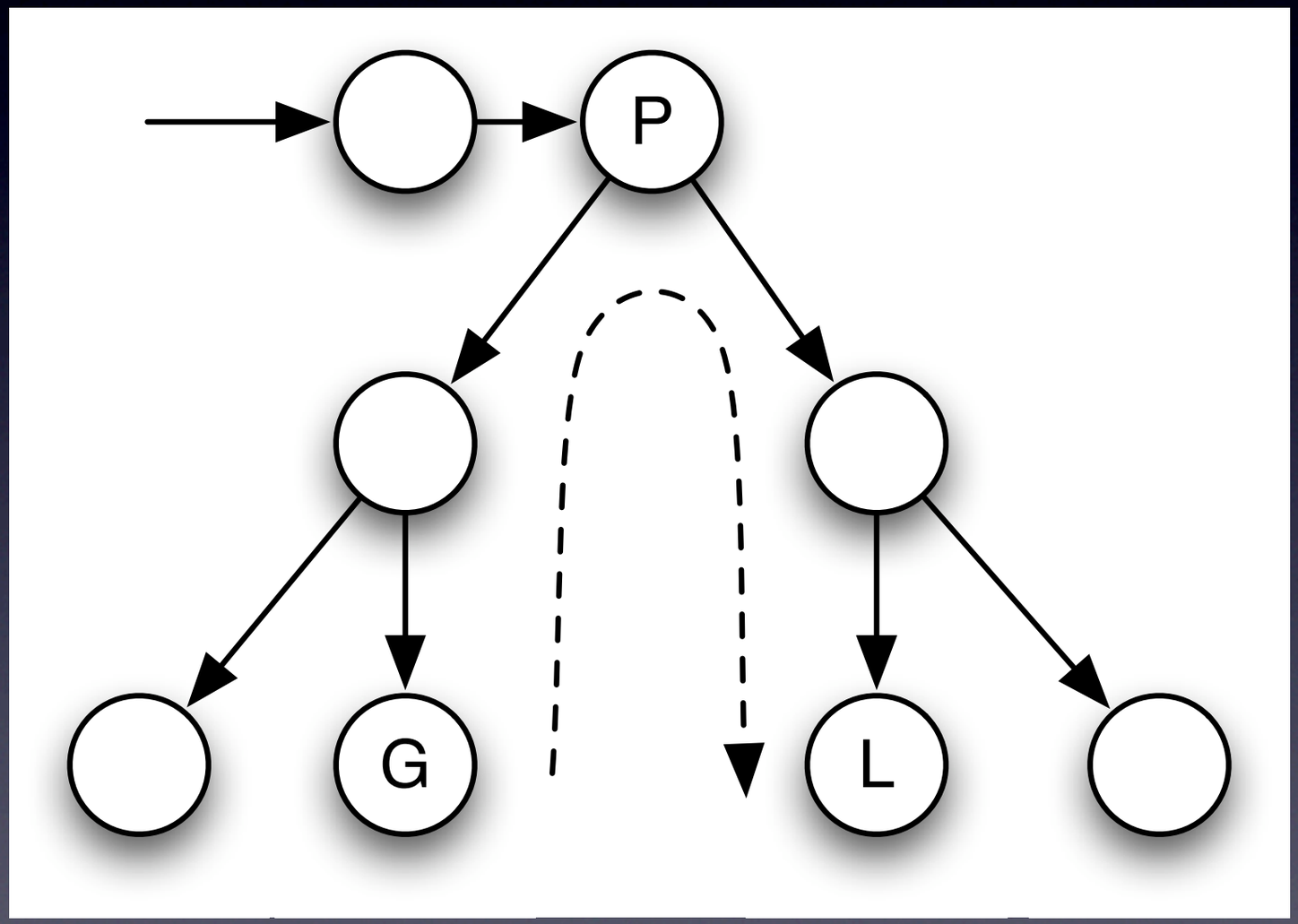
- Block
- If
- While

Nicht alles ist einfach



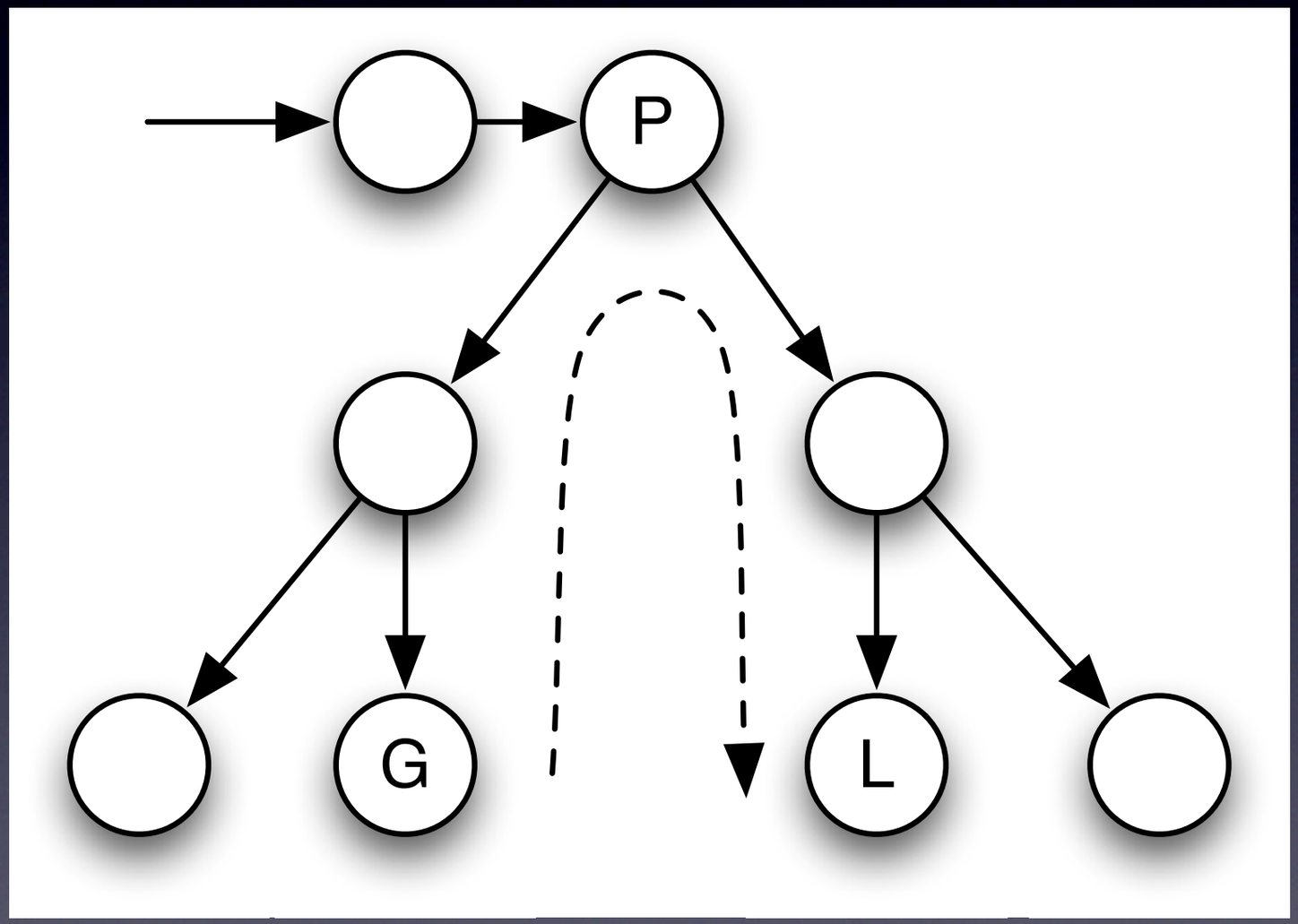
- Block
- If
- While
- Dowhile

Nicht alles ist einfach



- Block
- If
- While
- Dowhile
- For

Nicht alles ist einfach



- Block
- If
- While
- Dowhile
- For
- Switch

Auswärtsbewegungen

Block/If

Auswärtsbewegungen

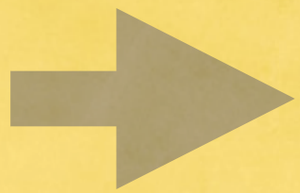
Block/If

```
{
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

Auswärtsbewegungen

Block/If

```
{
  /* ... */
  if (c) goto outward;
  /* ... */
}
```



Auswärtsbewegungen

Block/If

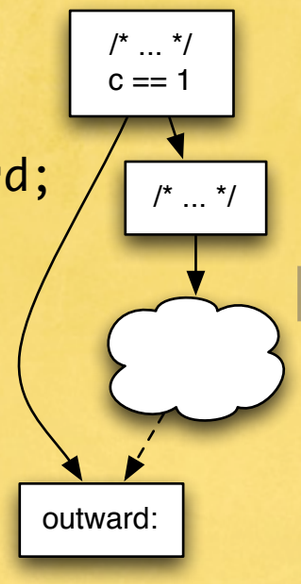
```
{
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

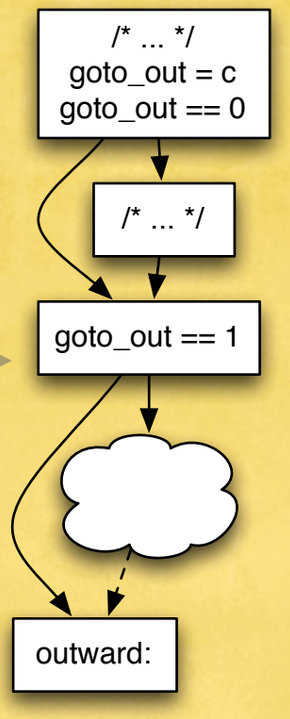
```
{
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
  }
}
if (goto_Outward)
  goto outward;
```

Auswärtsbewegungen

Block/If

```
{
  /* ... */
  if (c)
    goto outward;
  /* ... */
}
```






```
{
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
  }
}
if (goto_Outward)
  goto outward;
```

Auswärtsbewegungen

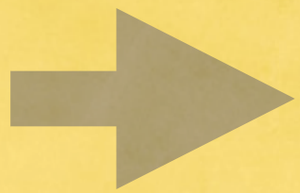
While

```
while (w) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

Auswärtsbewegungen

While

```
while (w) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```



Auswärtsbewegungen

While

```
while (w) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

```
while (!goto_Outward && w) {
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
  }
}
if (goto_Outward)
  goto outward;
```

Auswärtsbewegungen

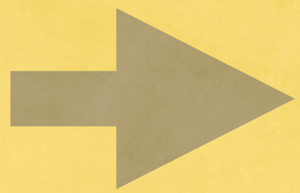
Dowhile

```
do {
  /* ... */
  if (c) goto outward;
  /* ... */
} while (w);
```

Auswärtsbewegungen

Dowhile

```
do {
  /* ... */
  if (c) goto outward;
  /* ... */
} while (w);
```



Auswärtsbewegungen

Dowhile

```
do {
  /* ... */
  if (c) goto outward;
  /* ... */
} while (w);
```

```
do {
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
  }
} while (!goto_Outward && w);
if (goto_Outward)
  goto outward;
```

Auswärtsbewegungen

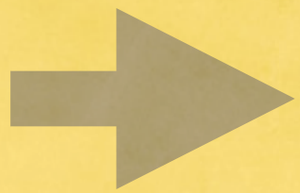
For

```
for (i; t; s) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

Auswärtsbewegungen

For

```
for (i; t; s) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```



Auswärtsbewegungen

For

```
for (i; t; s) {
  /* ... */
  if (c) goto outward;
  /* ... */
}
```

```
for (i; !goto_Outward && t;) {
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
    s
  }
}
if (goto_Outward)
  goto outward;
```

Auswärtsbewegungen

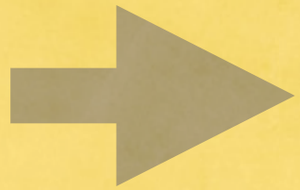
Switch

```
switch (/* ... */) {
/* ... */
case Y:
  /* ... */
  if (c) goto outward;
  /* ... */
  break;
  /* ... */
}
```

Auswärtsbewegungen

Switch

```
switch (/* ... */) {
/* ... */
case Y:
  /* ... */
  if (c) goto outward;
  /* ... */
  break;
  /* ... */
}
```



Auswärtsbewegungen

Switch

```
switch (/* ... */) {
/* ... */
case Y:
  /* ... */
  if (c) goto outward;
  /* ... */
  break;
  /* ... */
}
```

```
switch (/* ... */) {
/* ... */
case Y:
  /* ... */
  goto_Outward = c;
  if (!goto_Outward) {
    /* ... */
  }
  break;
  /* ... */
}
if (goto_Outward)
  goto outward;
```

Einwärtsbewegungen

Block/Dowhile

Einwärtsbewegungen

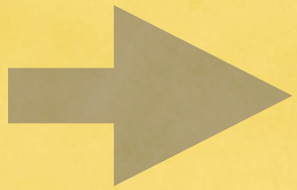
Block/Dowhile

```
if (c) goto inward;
/* ... */
{
  /* inward: is in here */
}
```

Einwärtsbewegungen

Block/Dowhile

```
if (c) goto inward;
/* ... */
{
  /* inward: is in here */
}
```



Einwärtsbewegungen

Block/Dowhile

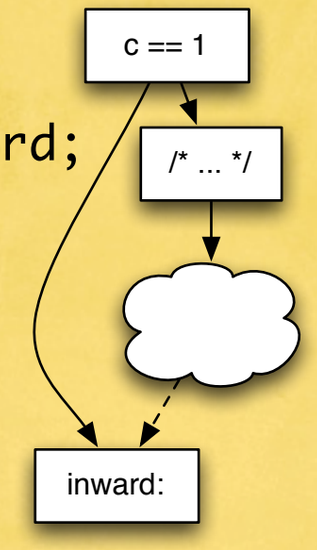
```
if (c) goto inward;
/* ... */
{
  /* inward: is in here */
}
```

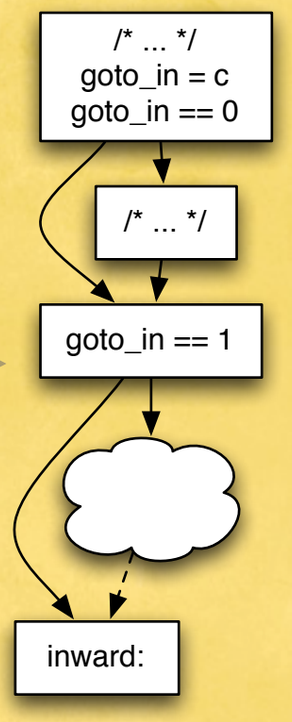
```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
}
{
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Einwärtsbewegungen

Block

```
if (c) goto inward;
/* ... */
{
  /* inward: */
}
```






```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
}
{
  if (goto_Inward)
    goto inward;
  /* inward: */
}
```

Einwärtsbewegungen

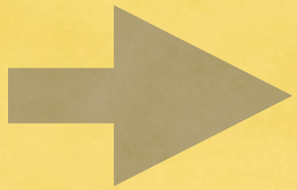
While

```
if (c) goto inward;
/* ... */
while (e) {
  /* inward: is in here */
}
```

Einwärtsbewegungen

While

```
if (c) goto inward;
/* ... */
while (e) {
  /* inward: is in here */
}
```



Einwärtsbewegungen

While

```
if (c) goto inward;
/* ... */
while (e) {
  /* inward: is in here */
}
```

```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
}
while (goto_Inward || e) {
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Einwärtsbewegungen

If

```
if (c) goto inward;
/* ... */
if (e) {
  /* inward: is in here */
}
```

```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
}
if (goto_Inward || e) {
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Einwärtsbewegungen

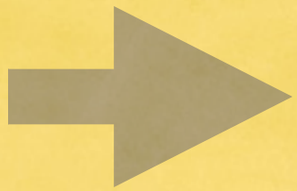
Else

```
if (c) goto inward;
/* ... */
if (e) {
  /* ... */
} else {
  /* inward: is in here */
}
```

Einwärtsbewegungen

Else

```
if (c) goto inward;
/* ... */
if (e) {
  /* ... */
} else {
  /* inward: is in here */
}
```



Einwärtsbewegungen

Else

```
if (c) goto inward;
/* ... */
if (e) {
  /* ... */
} else {
  /* inward: is in here */
}
```

```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
}
if (!goto_Inward && e) {
  /* ... */
} else {
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Einwärtsbewegungen

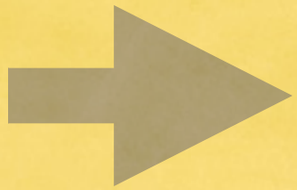
For

```
if (c) goto inward;
/* ... */
for (init; e; s) {
  /* inward: is in here */
}
```

Einwärtsbewegungen

For

```
if (c) goto inward;
/* ... */
for (init; e; s) {
  /* inward: is in here */
}
```



Einwärtsbewegungen

For

```
if (c) goto inward;
/* ... */
for (init; e; s) {
  /* inward: is in here */
}
```

```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
  init;
}
for (;
    goto_Inward || e; s) {
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Einwärtsbewegungen

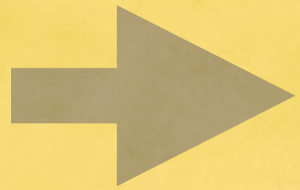
Switch

```
if (c) goto inward;
/* ... */
switch (e) {
case X:
  /* inward: is in here */
}
```

Einwärtsbewegungen

Switch

```
if (c) goto inward;
/* ... */
switch (e) {
case X:
  /* inward: is in here */
}
```



Einwärtsbewegungen

Switch

```
if (c) goto inward;
/* ... */
switch (e) {
case X:
  /* inward: is in here */
}
```

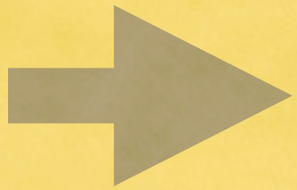
```
goto_Inward = c;
if (!goto_Inward) {
  /* ... */
  tmp = e;
} else {
  tmp = X;
}
switch (tmp) {
case X:
  if (goto_Inward)
    goto inward;
  /* inward: is in here*/
}
```

Lifting

```
{
  /* inward: is in here */
}
/* ... */
if (c) goto inward;
```

Lifting

```
{
  /* inward: is in here */
}
/* ... */
if (c) goto inward;
```



Lifting

```
{
  /* inward: is in here */
}
/* ... */
if (c) goto inward;
```

```
do {
  if (goto_Inward)
    goto inward;

  {
    /* inward: in here*/
  }

  /* ... */
  goto_Inward = c;
} while (goto_Inward);
```

Algorithmus

Algorithmus

- Flags für Labels einführen

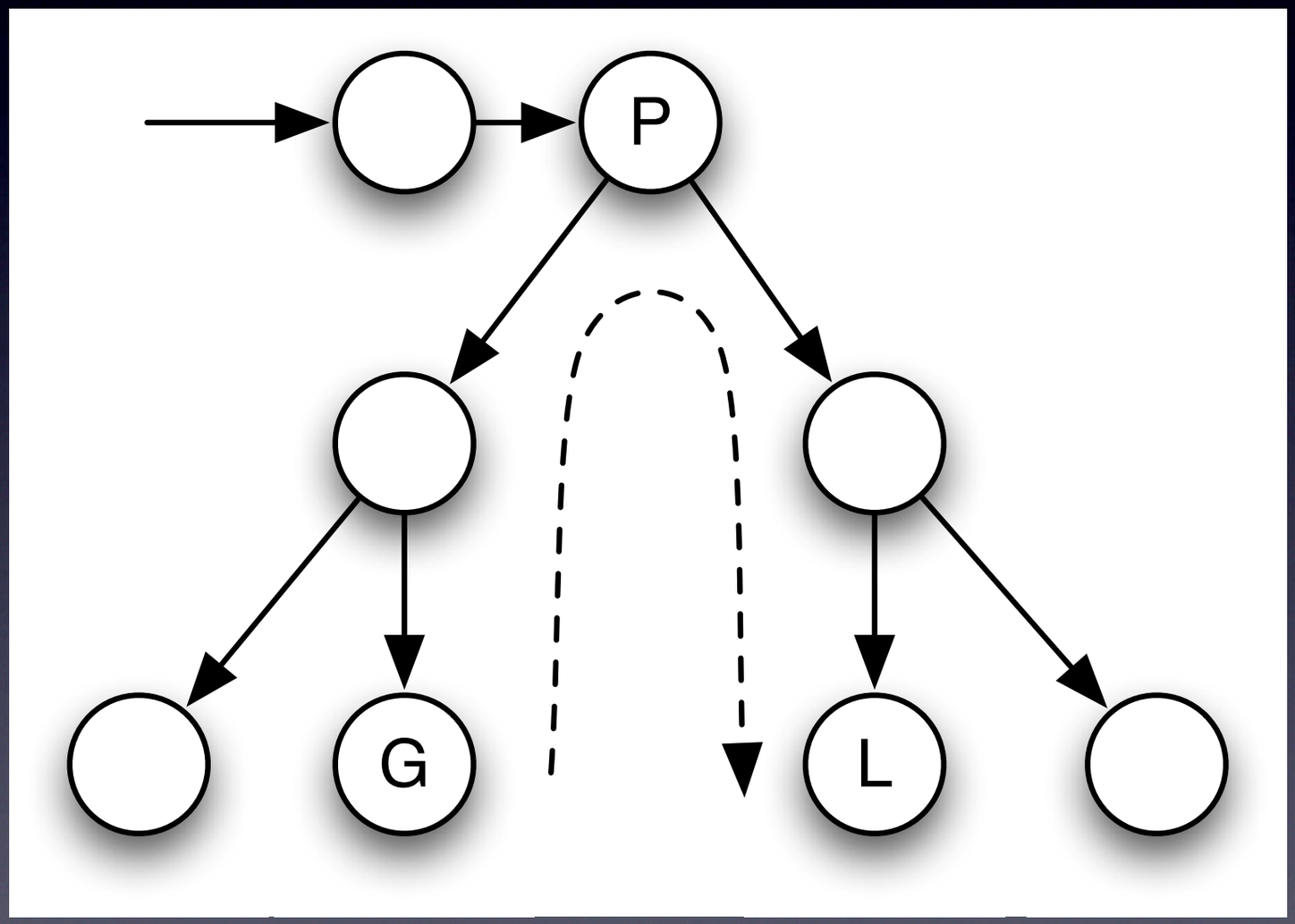
Algorithmus

- Flags für Labels einführen
- Bei Labelpositionen auf 0 setzen

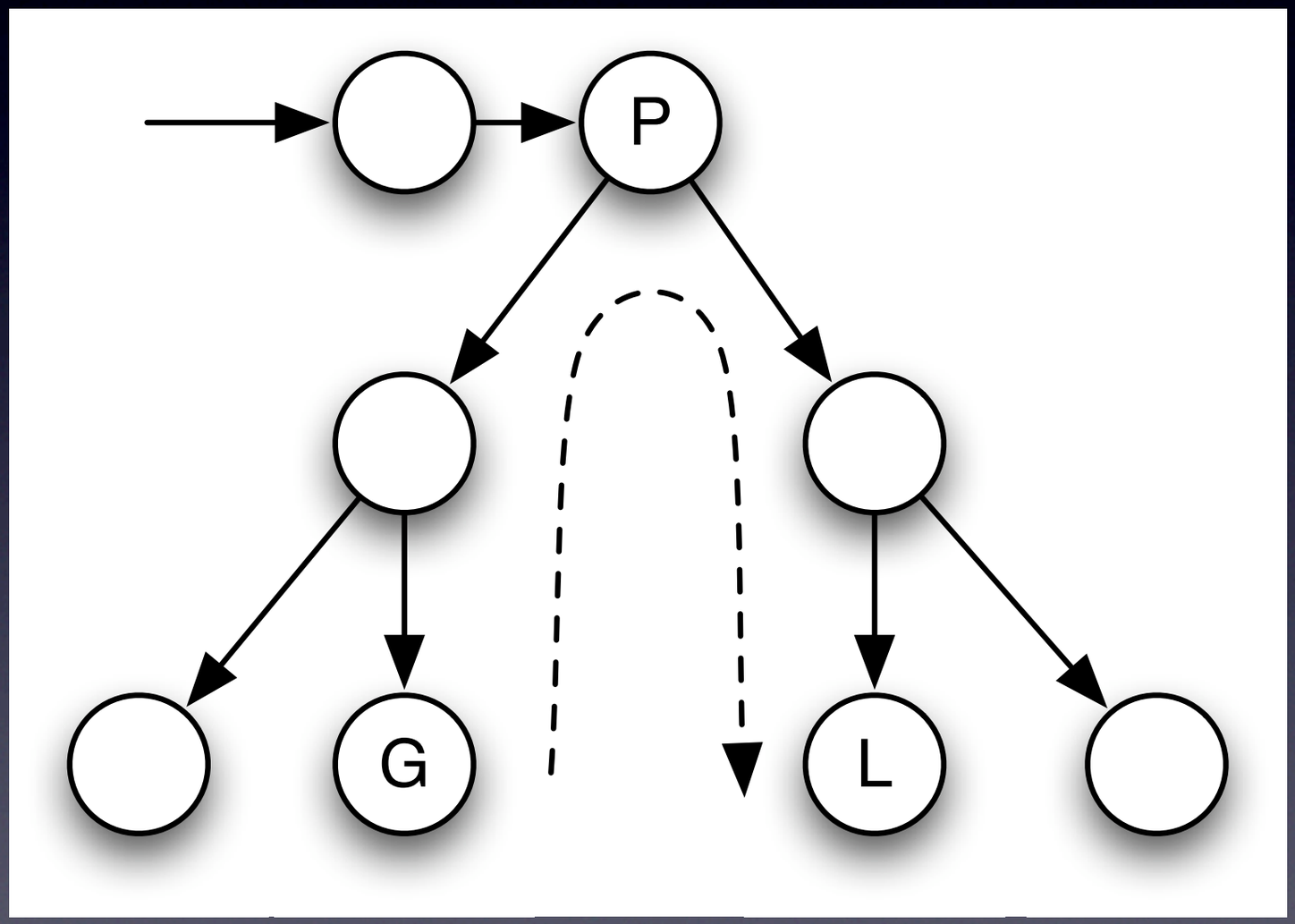
Algorithmus

- Flags für Labels einführen
- Bei Labelpositionen auf 0 setzen
- Goto/Label-Paare nacheinander entfernen

Main Loop



Main Loop




```
if (e) {
  goto label;
} else {
  /* label: is in here */
}
```

Beispiel

```
while (c) {
  /* ... */
  if (d) break;
}
```

Beispiel

```
while (c) {
  /* ... */
  if (d) goto out;
}
out:
```

Beispiel

```
goto_out = 0;

while (c) {
  /* ... */
  if (d) goto out;
}
out:
goto_out = 0;
```

Beispiel

```
goto_out = 0;

while (c) {
  /* ... */
  if (d) if (1) goto out;
}
out:
goto_out = 0;
```

Beispiel

```
goto_out = 0;

while (c) {
  /* ... */
  if (d) {
    goto_out = 1;
    if (!goto_out) {}
  }
  if (goto_out) goto out;
}
out:
goto_out = 0;
```

Beispiel

```
goto_out = 0;

while (!goto_out && c) {
  /* ... */
  if (d) {
    goto_out = 1;
    if (!goto_out) {}
  }
  goto_out = goto_out;
  if (!goto_out) { }
}
goto_out = 0;
```

AST-Cleanup

```
goto_out = 0;

while (!goto_out && c) {
  /* ... */
  if (d) {
    goto_out = 1;
    if (!goto_out) {}
  }
  goto_out = goto_out;
  if (!goto_out) { }
}
goto_out = 0;
```

AST-Cleanup

```
goto_out = 0;

while (!goto_out && c) {
  /* ... */
  if (d) {
    goto_out = 1;
  }
}
goto_out = 0;
```

AST-Cleanup

AST-Cleanup

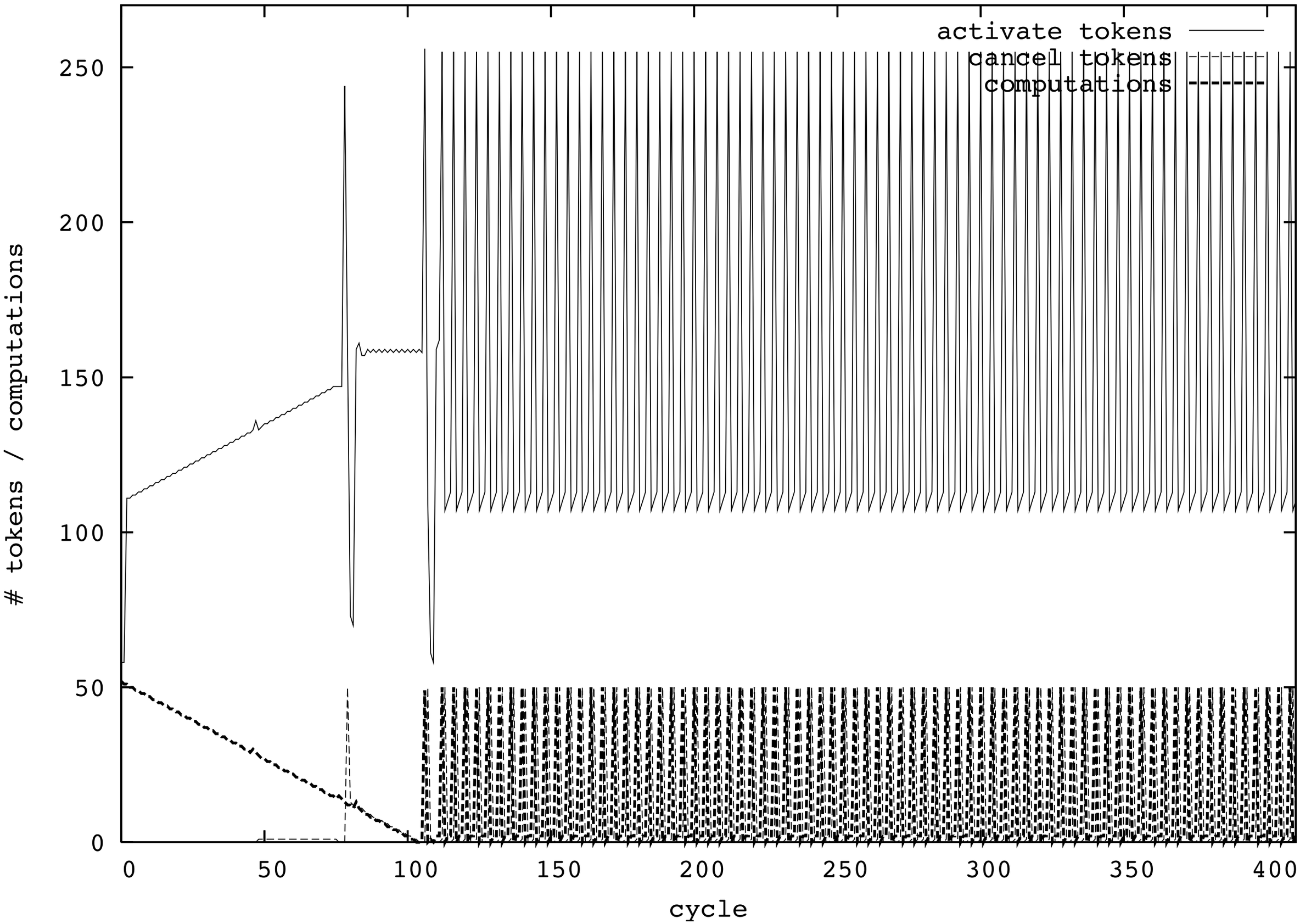
- Leere Ifs mit seiteneffektfreien Bedingungen

AST-Cleanup

- Leere Ifs mit seiteneffektfreien Bedingungen
- a=a-Zuweisungen

Beispiel mit Hardware

```
i = 0, a = 0;
n = 100;
while (i < n) {
  a += i;
  if (a != 0) {
    b0 += i;
    b1 += i;
    /* ... */
    b49 +=i;
  }
  ++i;
}
```

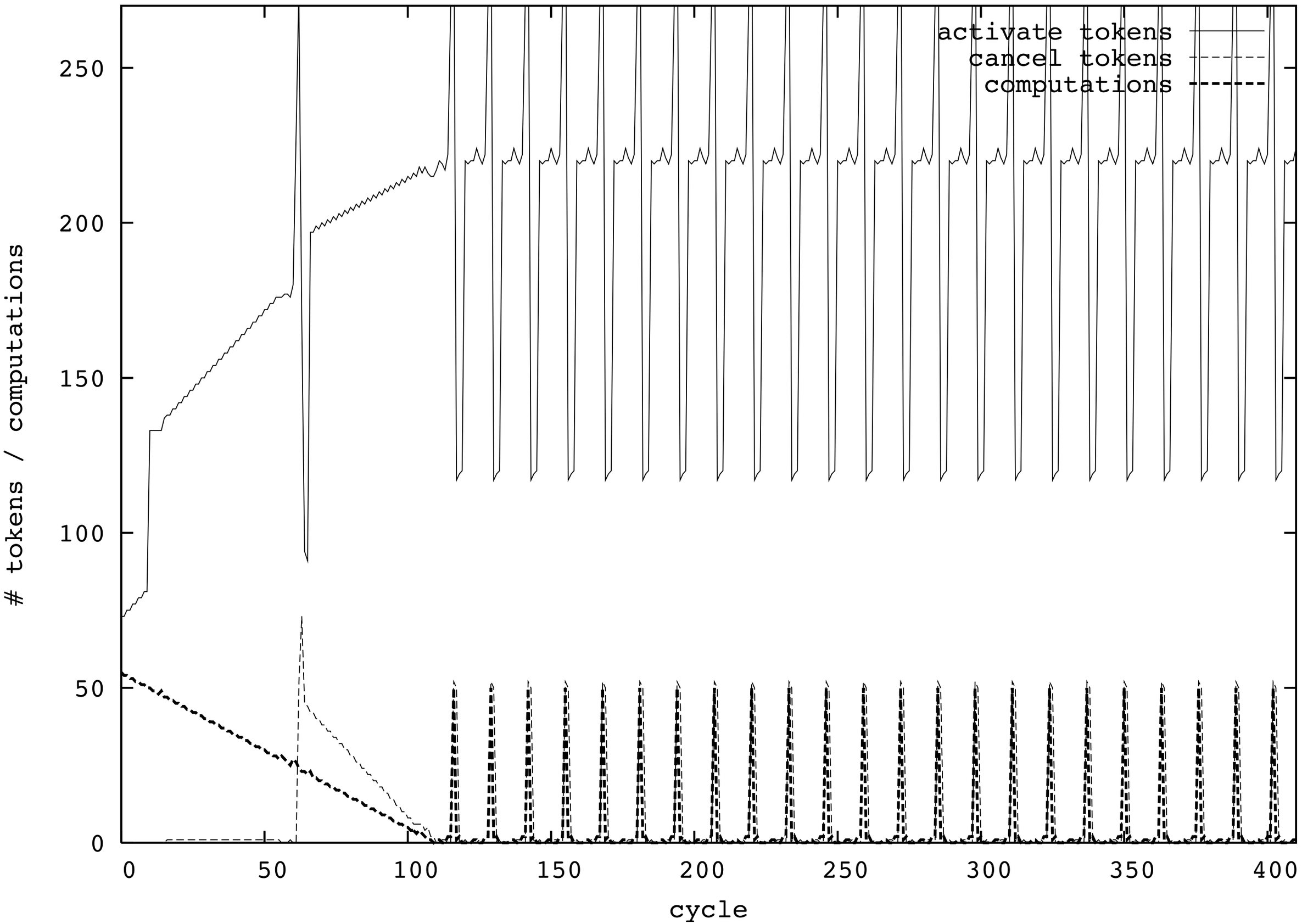

activate tokens
cancel tokens
computations
tokens / computations
250
200
150
100
50
0
0
50
100
150
200
250
300
350
400
cycle

Beispiel mit Hardware

```
i = 0, a = 0;
n = 100;
while (i < n) {
  a += i;
  if (a > 400) break;
  if (a != 0) {
    b0 += i;
    b1 += i;
    /* ... */
    b49 +=i;
  }
  ++i;
}
```

Beispiel mit Hardware

```
i = 0, a = 0;
n = 100;
goto_out = 0;
while (!goto_out && i < n) {
  a += i;
  if (a > 400) goto_out = 1;
  if (!goto_out) {
    if (a != 0) {
      b0 += i;
      b1 += i;
      /* ... */
      b49 +=i;
    }
    ++i;
  }
}
```

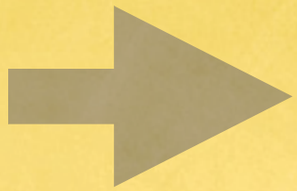

activate tokens
cancel tokens
computations
tokens / computations
cycle
0
50
100
150
200
250
300
350
400

SUIF ist schuld

```
while (!goto_out && i < n) {
  /* ... */
}
```

SUIF ist schuld

```
while (!goto_out && i < n) {
  /* ... */
}
```



SUIF ist schuld

```
while (!goto_out && i < n) {
  /* ... */
}
```

```
while (runloop) {
  /* ... */
  runloop = 0;
  if (!goto_out) {
    is_bound_valid = (i < n);
    if (is_bound_valid) {
      runloop = 1;
    }
  }
}
```

Fertig

Das war’s

Zugabe

- Falls Zeit (muss mal probehalten) und Interesse, ein paar Folien über Effizienz (StatementList: Grrr), Tests/Build/Backtrace-Modul/Vim/Refactoring-Buch