Globale Optimierung von OSEK Systemen

durch “Analyse ohne (Kernel-)Grenzen”

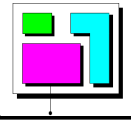
Christian Dietrich

Masterarbeit

Friedrich-Alexander Universität
Erlangen-Nürnberg

10. Oktober 2014

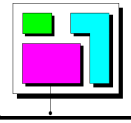
OSEK: Ein Standard mit Potential



- OSEK[1] ist ein Industriestandard aus der Automobilbranche
  - Erste Version von 1997 (Daimler, Mercedes-Benz, Bosch, Renault, . . .)
  - OSEK-OS: Betriebsystem-Schnittstelle für Echtzeitanwendungen

[1] **O**ffene **S**ysteme und deren Schnittstellen für die **E**lektronik in **K**raftfahrzeugen

OSEK: Ein Standard mit Potential

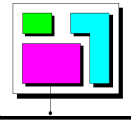


- OSEK[1] ist ein Industriestandard aus der Automobilbranche
  - Erste Version von 1997 (Daimler, Mercedes-Benz, Bosch, Renault, . . .)
  - OSEK-OS: Betriebsystem-Schnittstelle für Echtzeitanwendungen
- OSEK-OS ist ein statisches Betriebssystem
  - Anwendung kann nicht ausgetauscht werden
  - Explizite Deklarierung aller Systemobjekte (OIL)
  - Konstante Anzahl an Tasks; statisch zugewiesene Priorität
  - Alle Unterbrecher und Alarme sind vorher bekannt

$\Rightarrow$ Viel statisches Wissen verfügbar

[1] **O**ffene **S**ysteme und deren Schnittstellen für die **E**lektronik in **K**raftfahrzeugen

OSEK: Ein Standard mit Potential

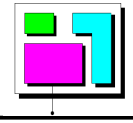


- OSEK[1] ist ein Industriestandard aus der Automobilbranche
  - Erste Version von 1997 (Daimler, Mercedes-Benz, Bosch, Renault, . . .)
  - OSEK-OS: Betriebsystem-Schnittstelle für Echtzeitanwendungen
- OSEK-OS ist ein statisches Betriebssystem
  - Anwendung kann nicht ausgetauscht werden
  - Explizite Deklarierung aller Systemobjekte (OIL)
  - Konstante Anzahl an Tasks; statisch zugewiesene Priorität
  - Alle Unterbrecher und Alarme sind vorher bekannt

  $\Rightarrow$ Viel statisches Wissen verfügbar
- OSEK profitiert bereits von statischer Maßschneiderung
  - Statische Allokation von Systemobjekten
  - Statische Initialisierung
  - Automatische Auswahl von Systemkomponenten

[1] **O**ffene **S**ysteme und deren Schnittstellen für die **E**lektronik in **K**raftfahrzeugen

OSEK: Ein Standard mit Potential



- OSEK[1] ist ein Industriestandard aus der Automobilbranche
  - Erste Version von 1997 (Daimler, Mercedes-Benz, Bosch, Renault, . . .)
  - OSEK-OS: Betriebsystem-Schnittstelle für Echtzeitanwendungen
- OSEK profitiert bereits von statischer Maßschneiderung
  - Statische Allokation von Systemobjekten
  - Statische Initialisierung
  - Automatische Auswahl von Systemkomponenten

  - Anwendungswissen erlaubt statische Maßschneiderung
  - Grobgranulare Anpassung durch OIL-Wissen
  - Feingranulare Anpassung erfordert feingranulares Wissen
  - Weitere Optimierung von Laufzeit oder Fehlertoleranz

[1] **O**ffene **S**ysteme und deren Schnittstellen für die **E**lektronik in **K**raftfahrzeugen

- **Frage 1:** Wie erhalte ich dieses feingranulares Interaktionswissen?

- **Frage 2:** Wie nutze dieses Wissen?

- **Frage 1:** Wie erhalte ich dieses feingranulares Interaktionswissen?

  ↳ Der globale Kontrollflusgraph

- **Frage 2:** Wie nutze dieses Wissen?

Ein vereinfachtes OSEK System

Task 1; Priorität 4

Task 2; Priorität 5

Task 3; Priorität 3

Ein vereinfachtes OSEK System

Task 1; Priorität 4

```
TASK(Task1) {
    int data = read_data();
    if (data == '\0') {
      ActivateTask(Task3);
    } else {
      bb_put(data);
    }
    ChainTask(Task2);
}
```

Task 2; Priorität 5

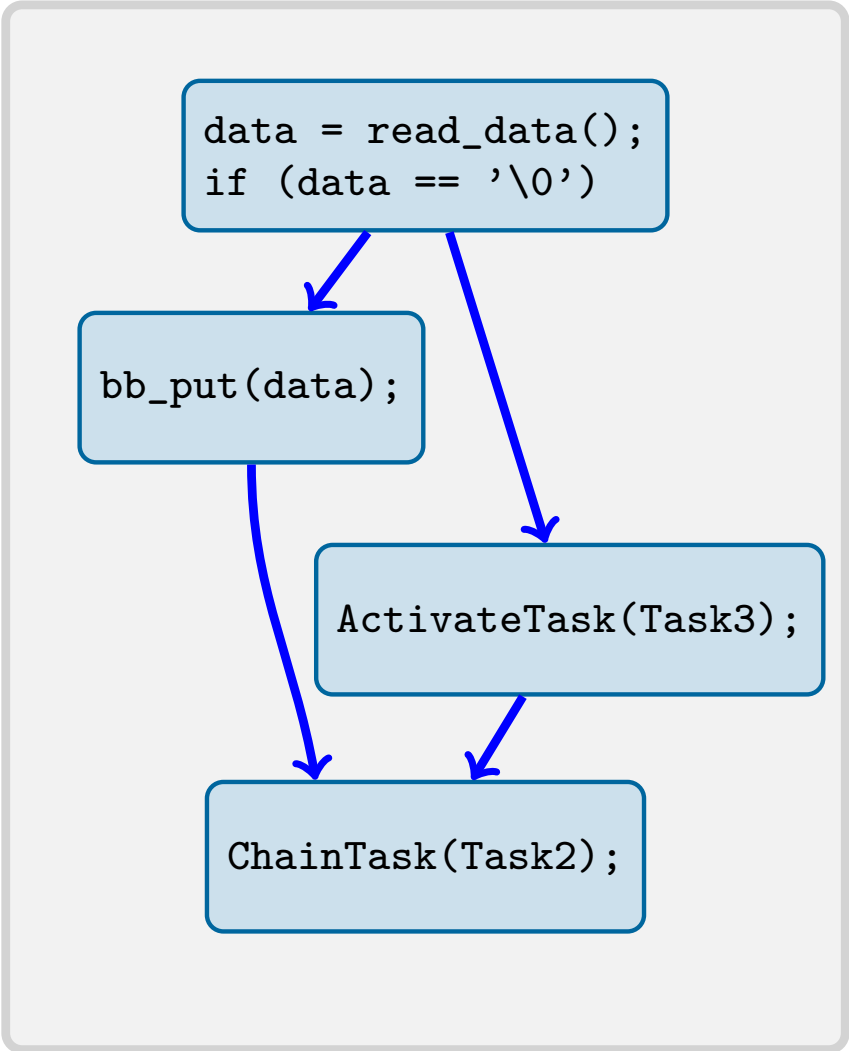
```
TASK(Task2) {

  setup_of_device();

  TerminateTask();
}
```

Task 3; Priorität 3

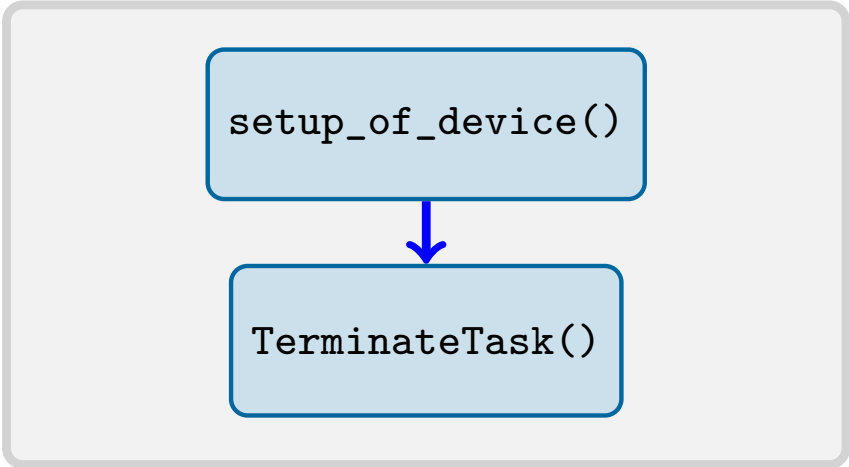
```
TASK(Task3) {
  parse_message();
  bb_clear_buffer();

  TerminateTask();
}
```

Der Kontrollflussgraph (CFG)

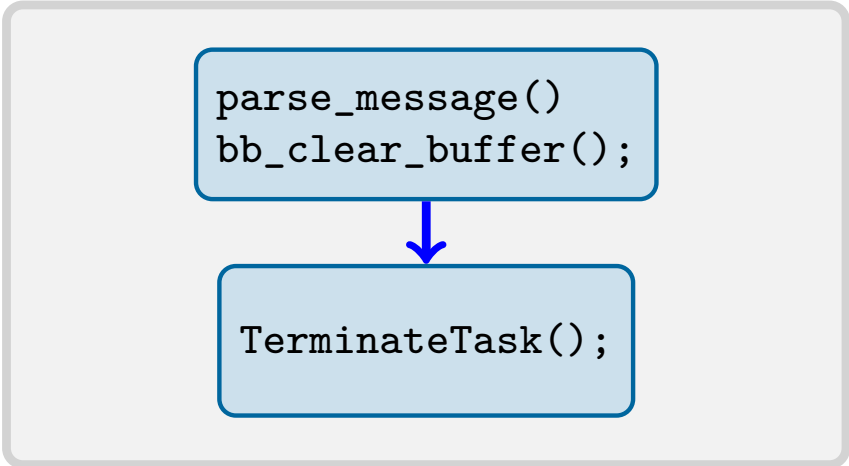
Task 1; Priorität 4



Task 2; Priorität 5

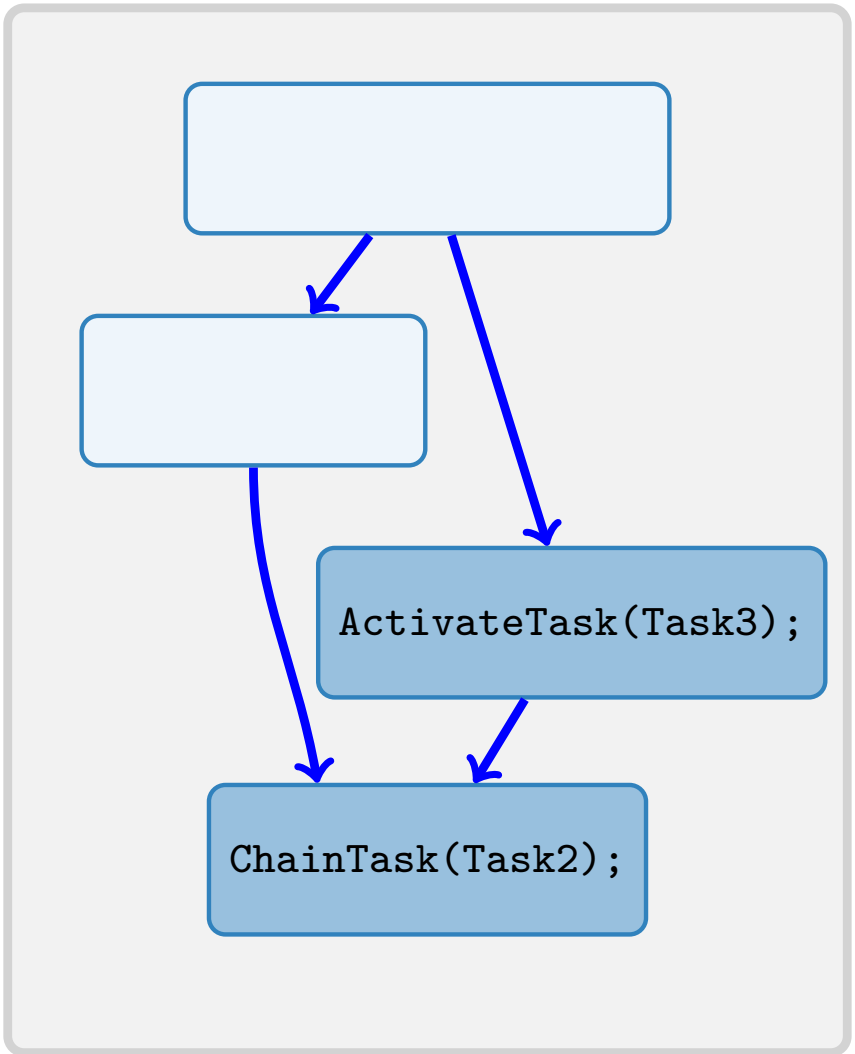


Task 3; Priorität 3

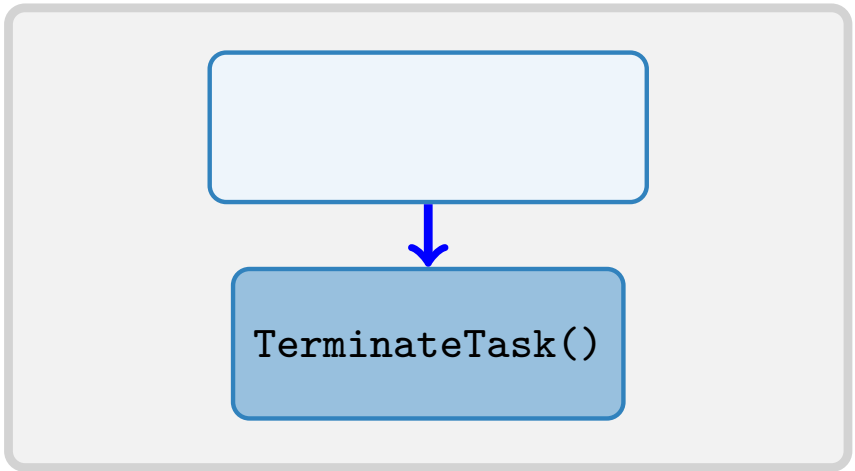


Der Kontrollflussgraph (CFG)

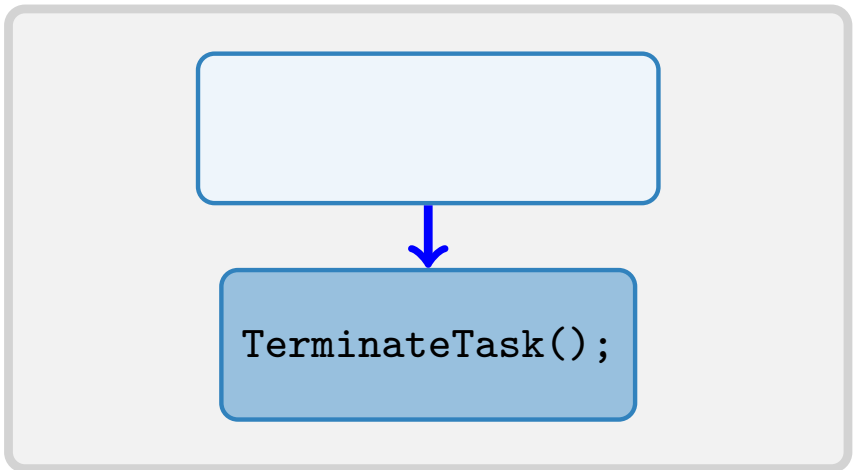
Task 1; Priorität 4



Task 2; Priorität 5



Task 3; Priorität 3



Der globale Kontrollflussgraph (GCFG)

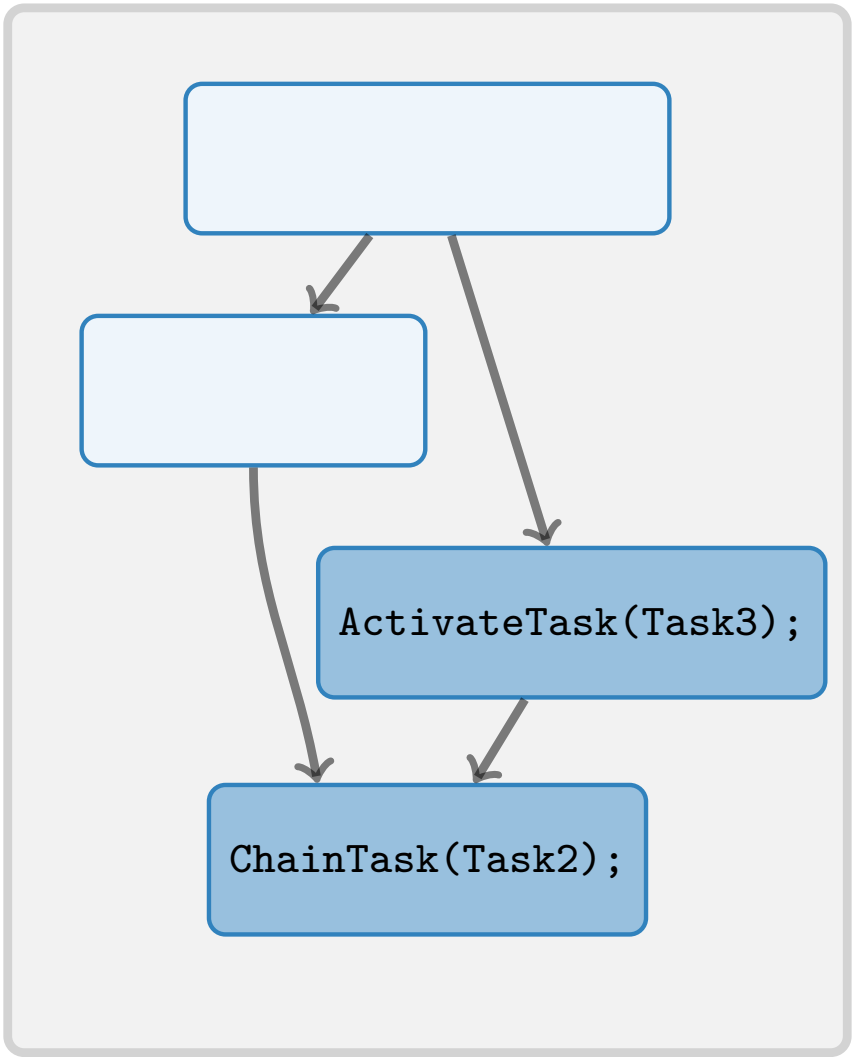
- Jeder Task hat einen eigenen Kontrollflussgraphen
  - CFG ist nur task-lokal
  - Das Betriebssystem stellt “virtuellen” Prozessor bereit
  - CFG := Alle möglichen Block-Sequenzen auf virtuellem Prozessor

Der globale Kontrollflussgraph (GCFG)

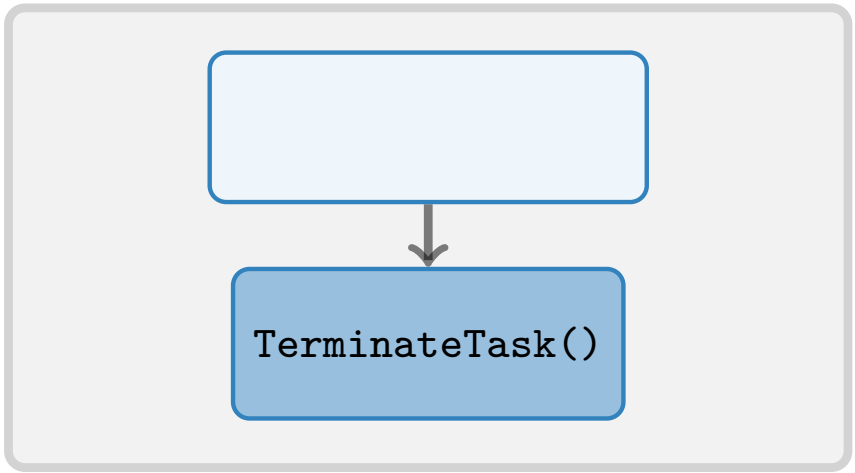
- Jeder Task hat einen eigenen Kontrollflussgraphen
  - CFG ist nur task-lokal
  - Das Betriebssystem stellt “virtuellen” Prozessor bereit
  - CFG := Alle möglichen Block-Sequenzen auf virtuellem Prozessor
- Jedes System hat einen globalen Kontrollflussgraphen (GCFG)
  - Der GCFG ist system-global
  - Das Betriebssystem hat nur einen realen Prozessor
  - GCFG := Alle möglichen Block-Sequenzen auf realem Prozessor

Der Globale Kontrollflussgraph (GCFG)

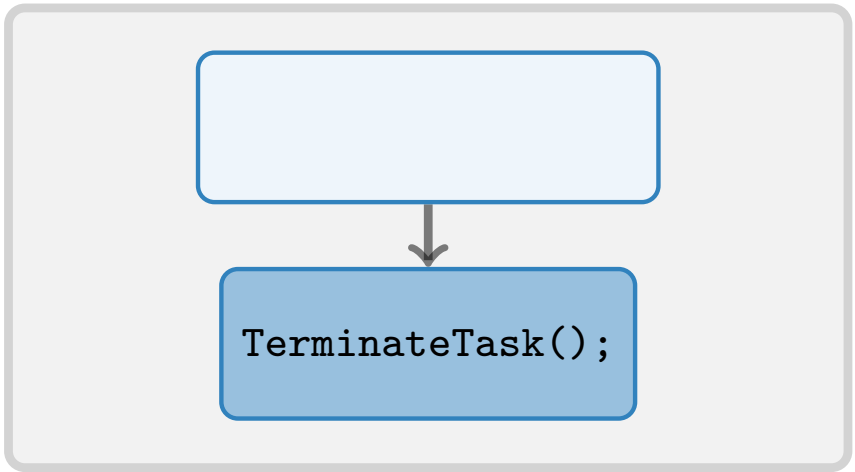
Task 1; Priorität 4



Task 2; Priorität 5

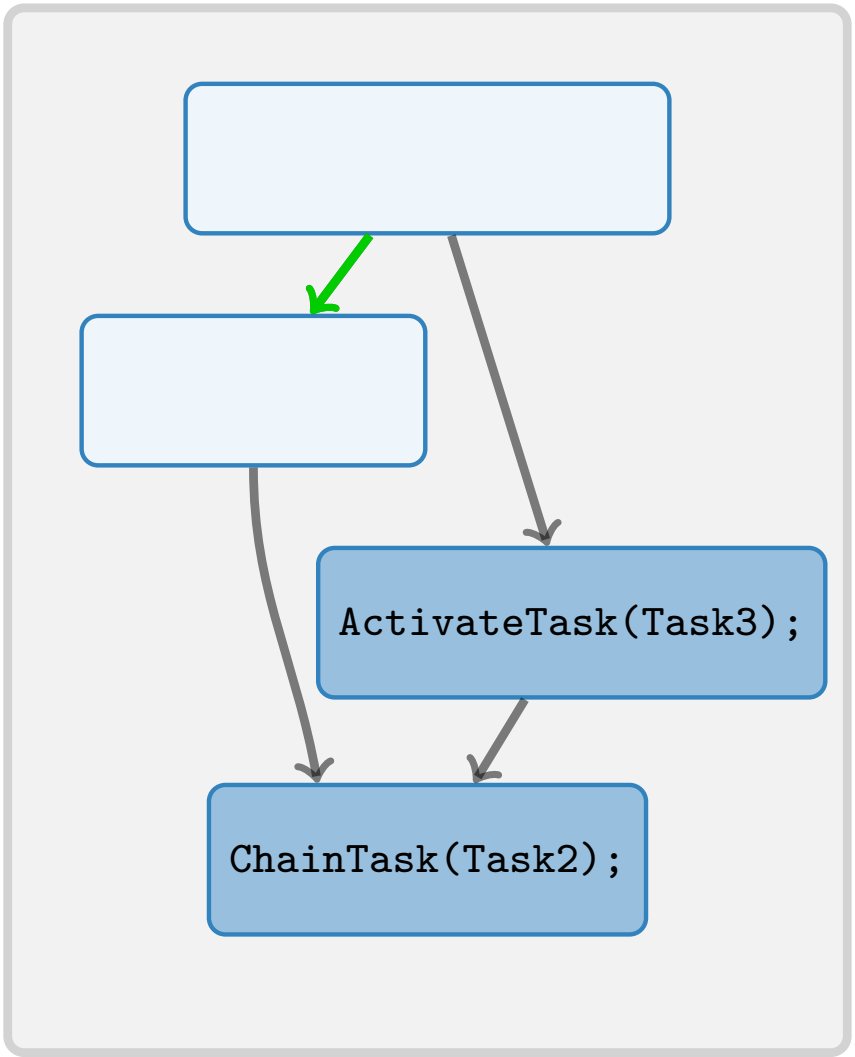


Task 3; Priorität 3

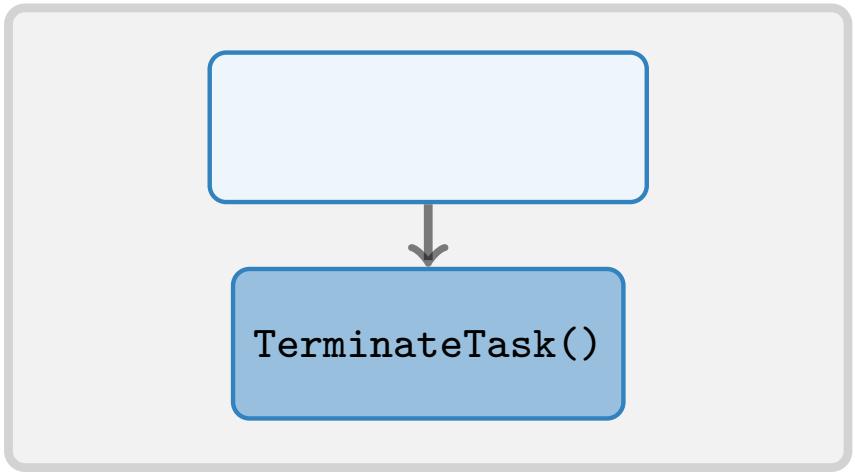


Der Globale Kontrollflussgraph (GCFG)

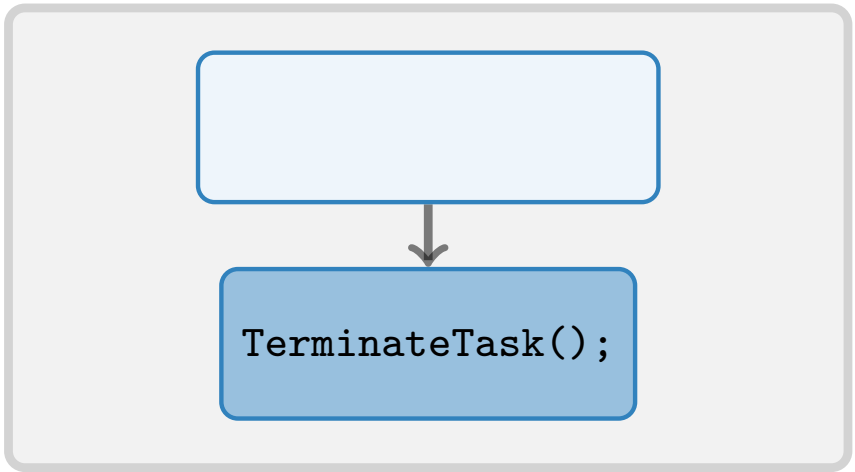
Task 1; Priorität 4



Task 2; Priorität 5

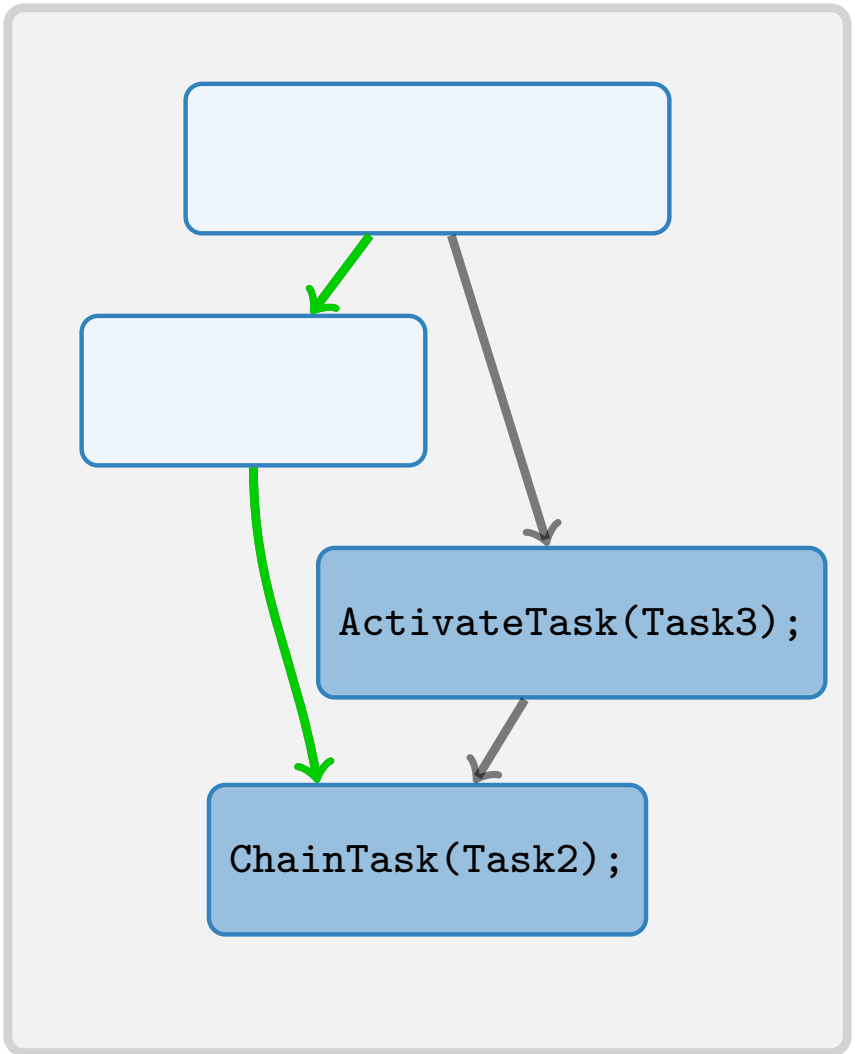


Task 3; Priorität 3

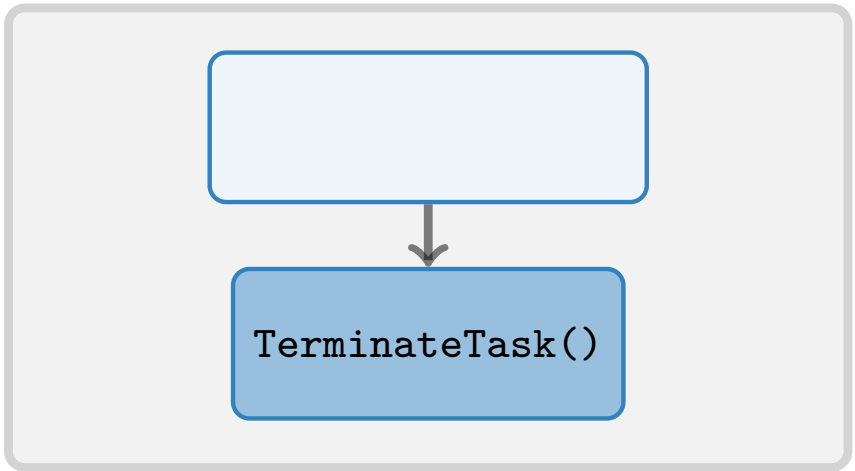


Der Globale Kontrollflussgraph (GCFG)

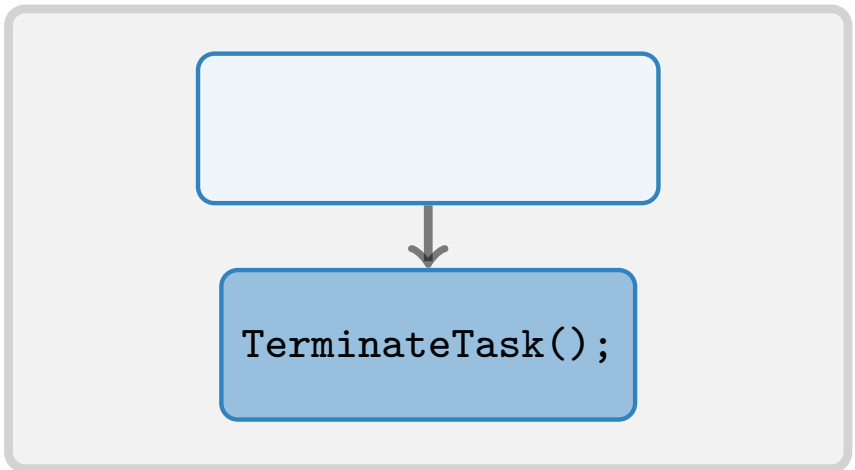
Task 1; Priorität 4



Task 2; Priorität 5

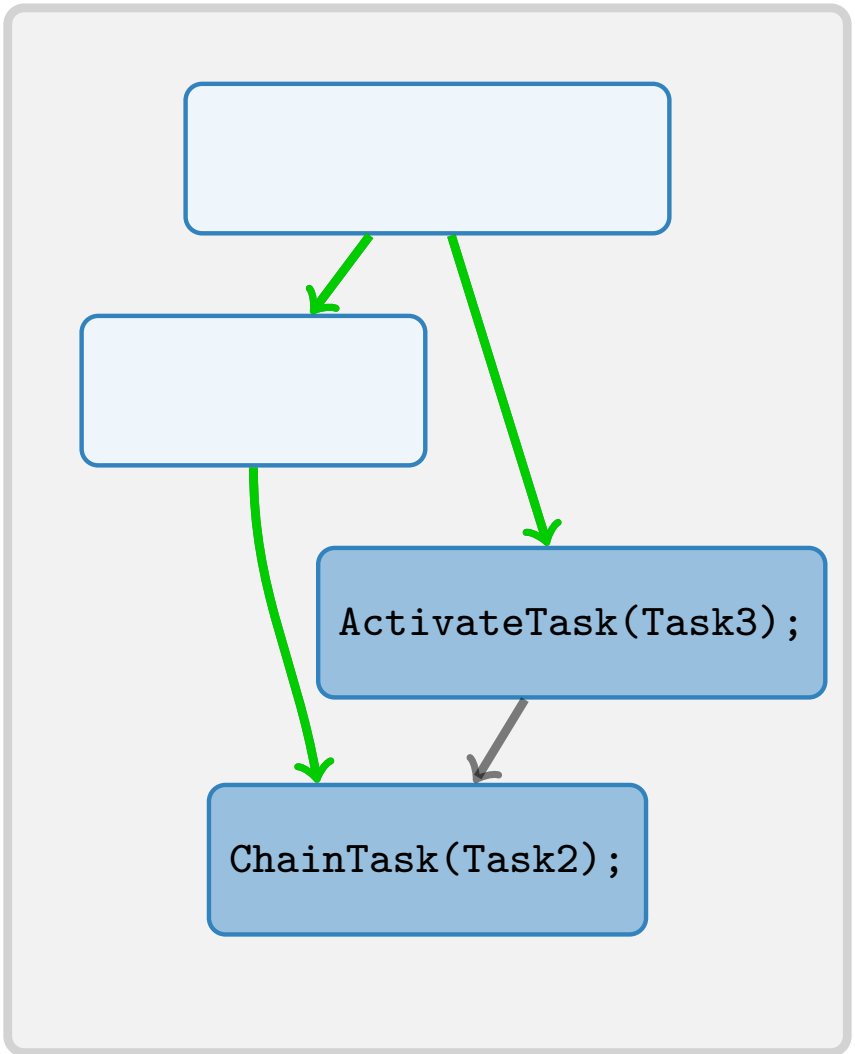


Task 3; Priorität 3

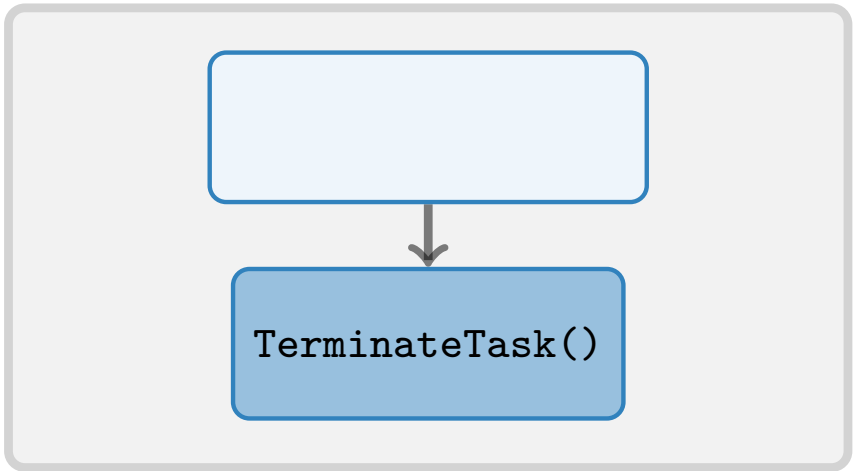


Der Globale Kontrollflussgraph (GCFG)

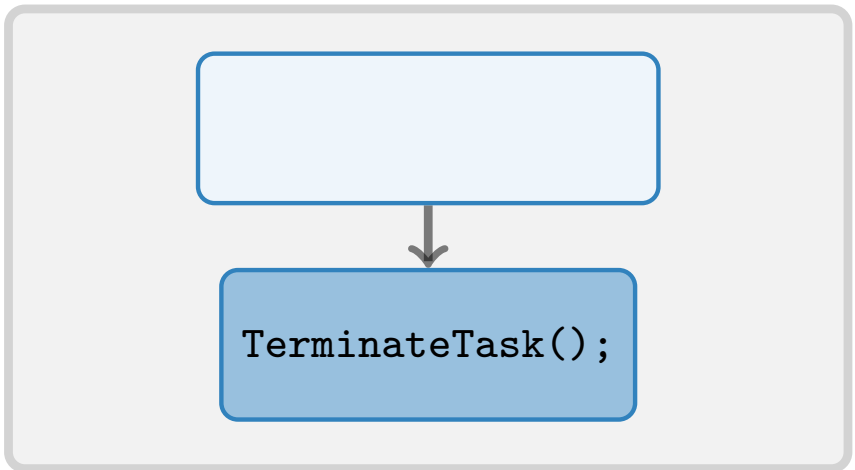
Task 1; Priorität 4



Task 2; Priorität 5

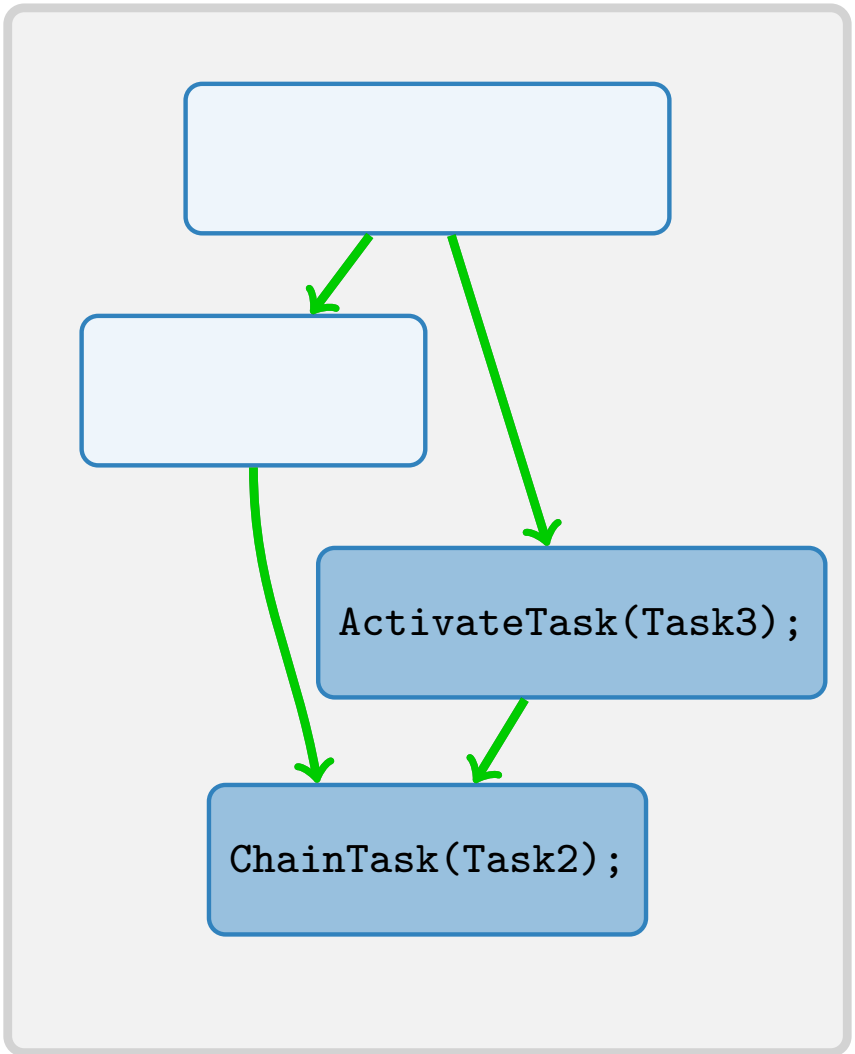


Task 3; Priorität 3

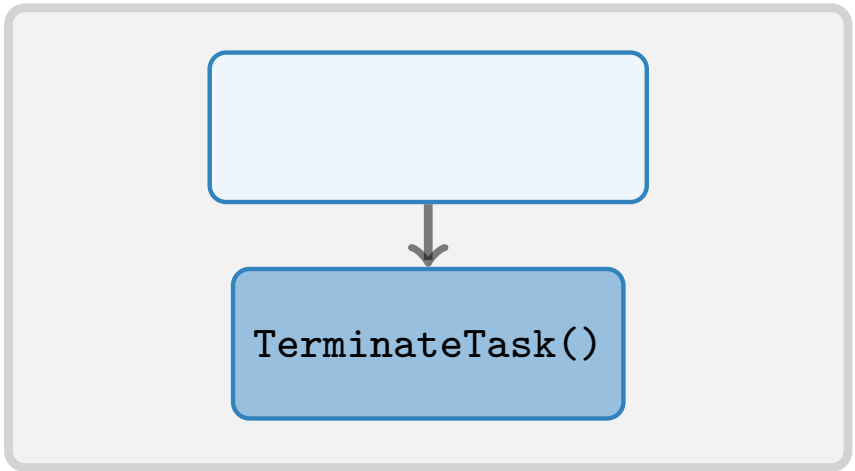


Der Globale Kontrollflussgraph (GCFG)

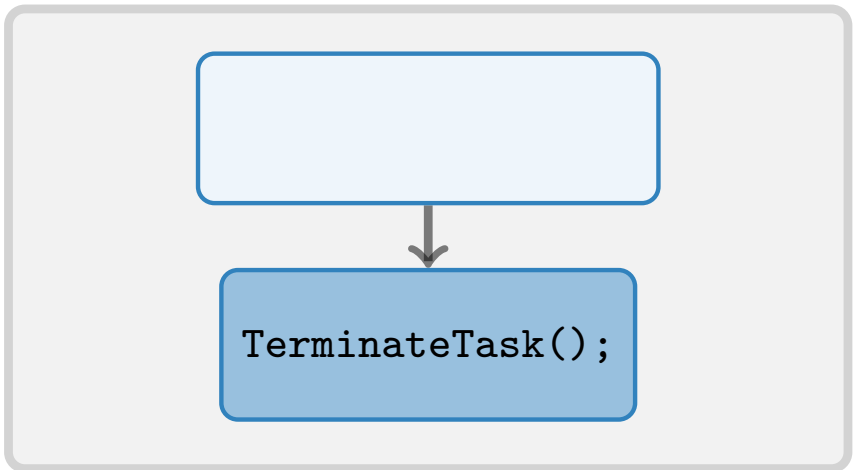
Task 1; Priorität 4



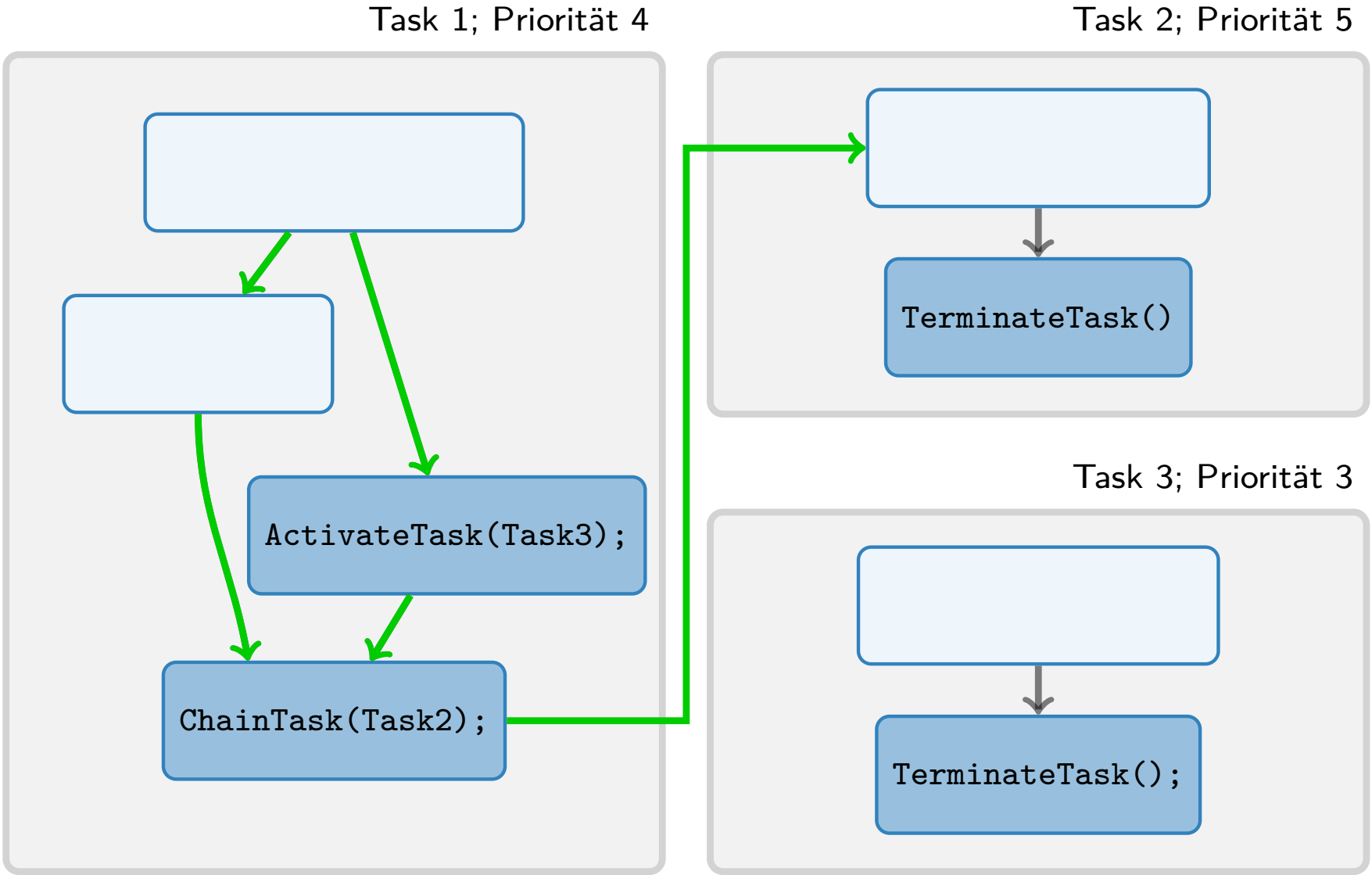
Task 2; Priorität 5



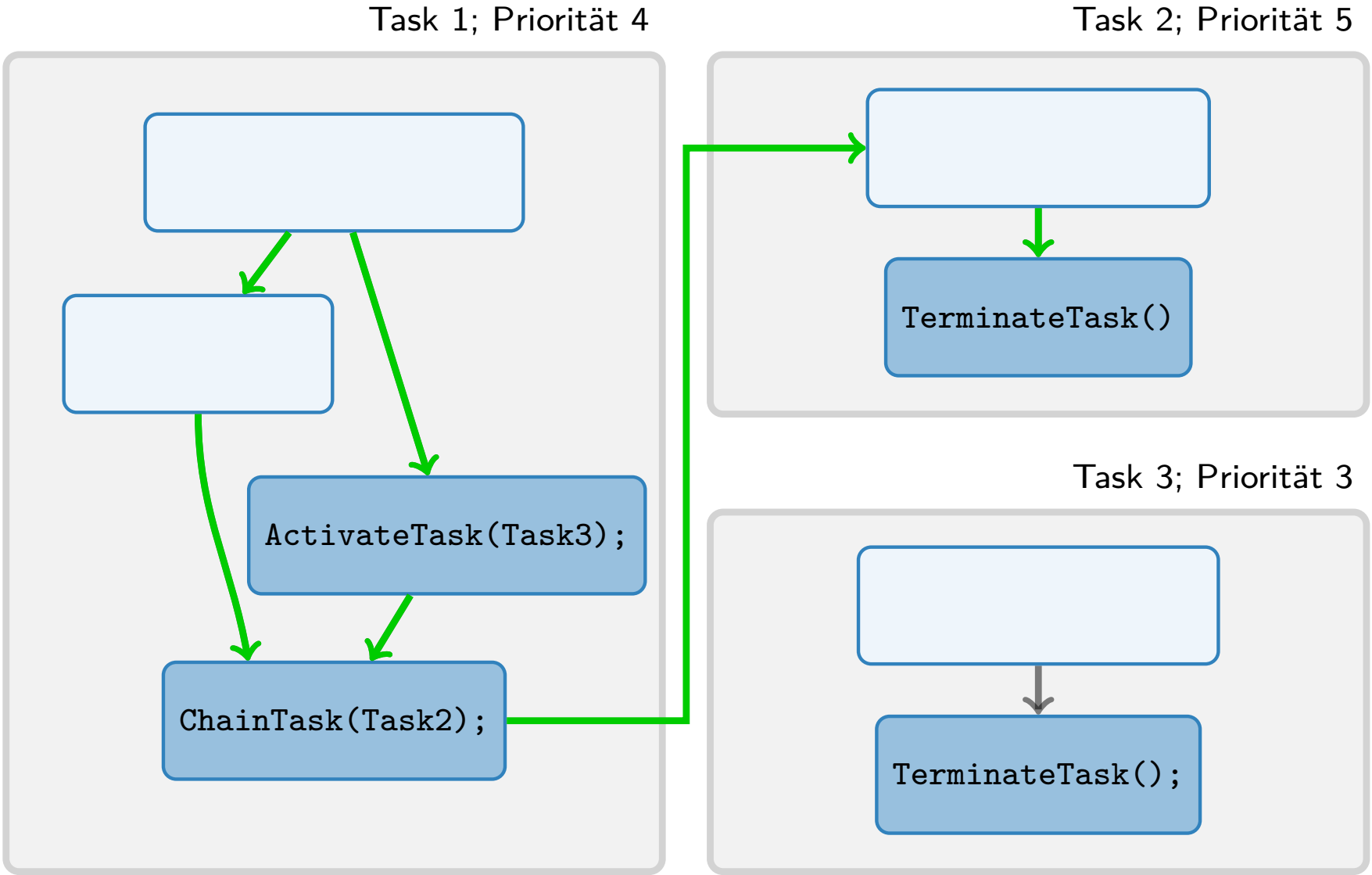
Task 3; Priorität 3



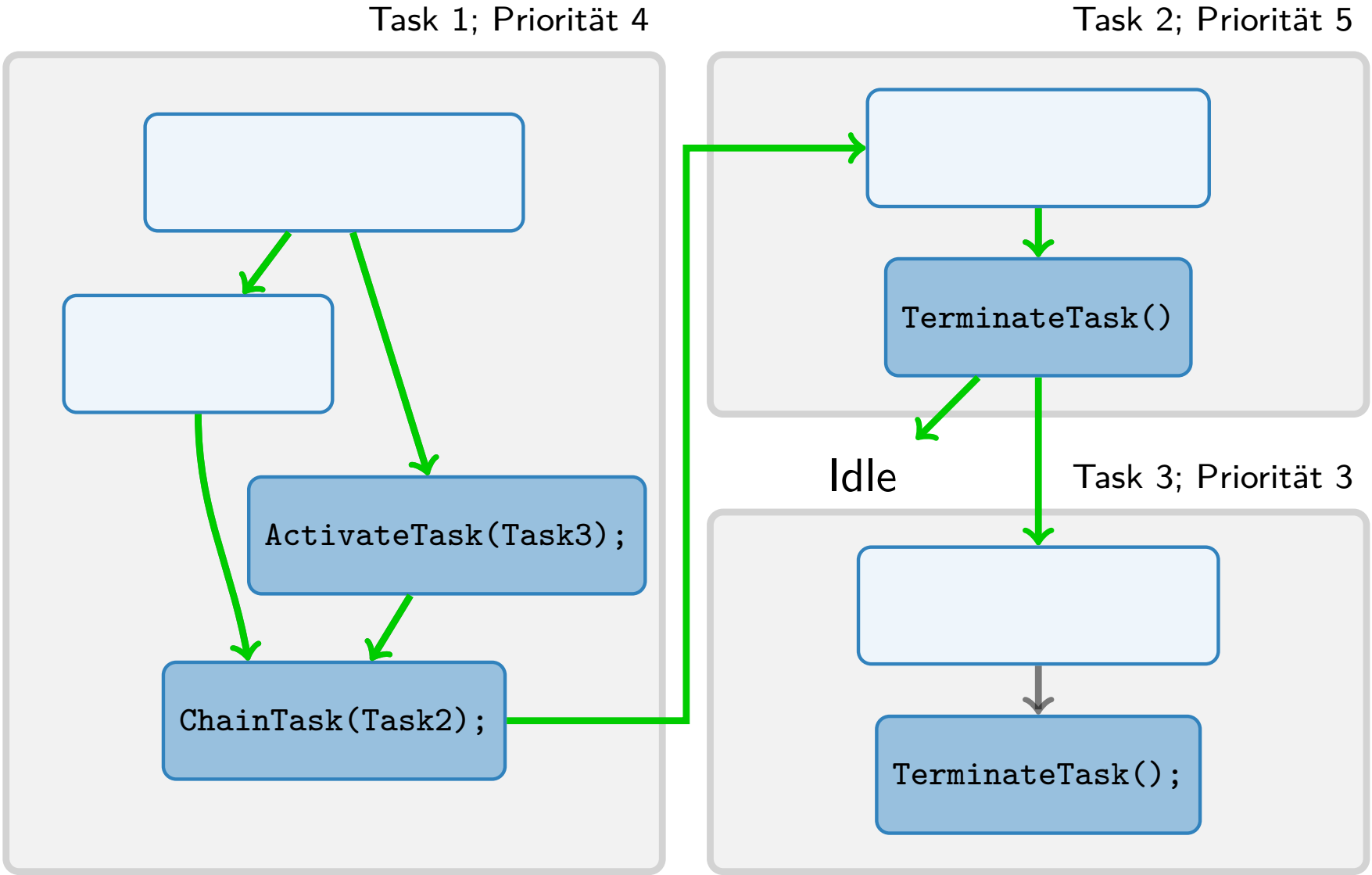
Der Globale Kontrollflussgraph (GCFG)



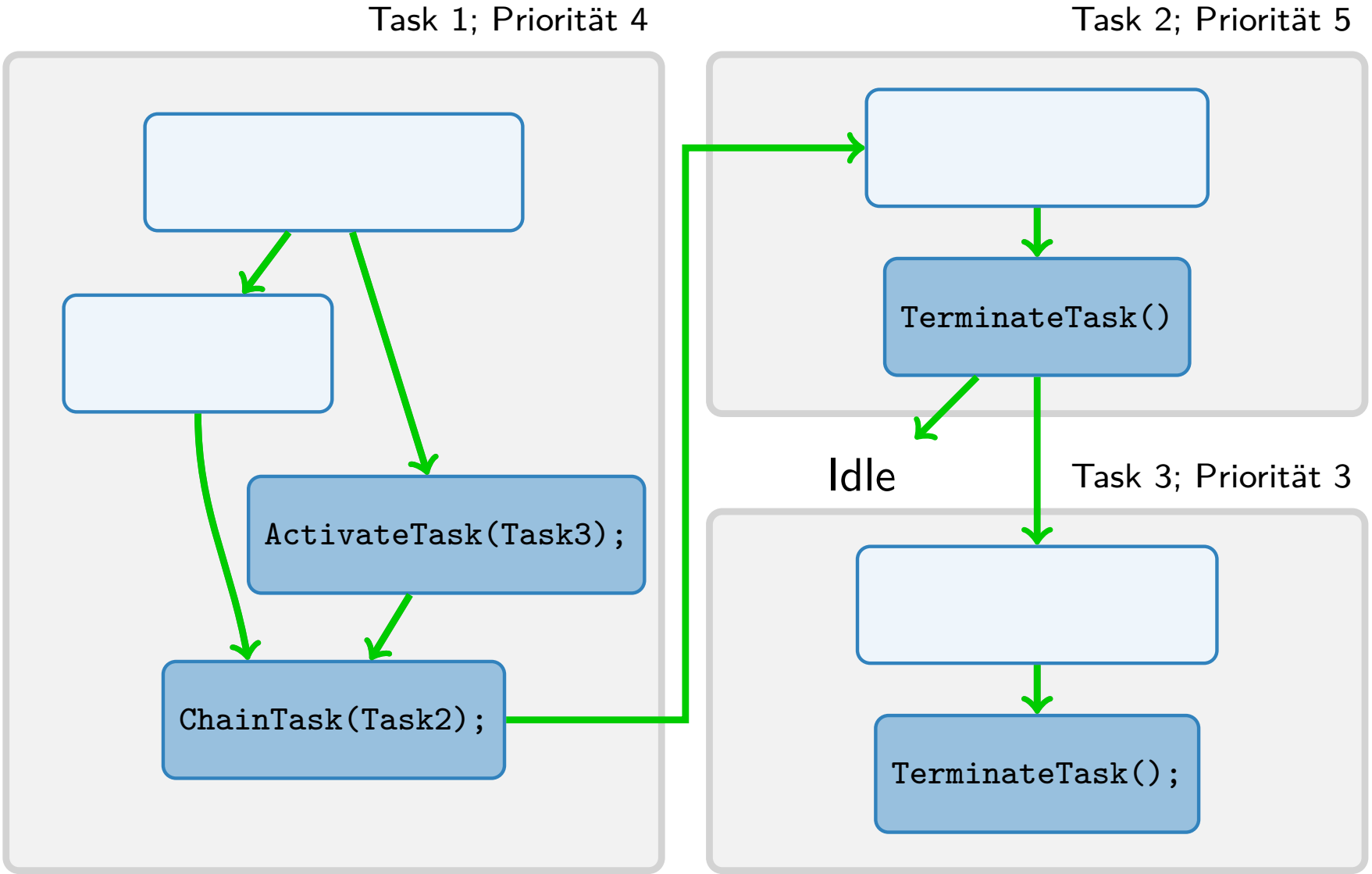
Der Globale Kontrollflussgraph (GCFG)



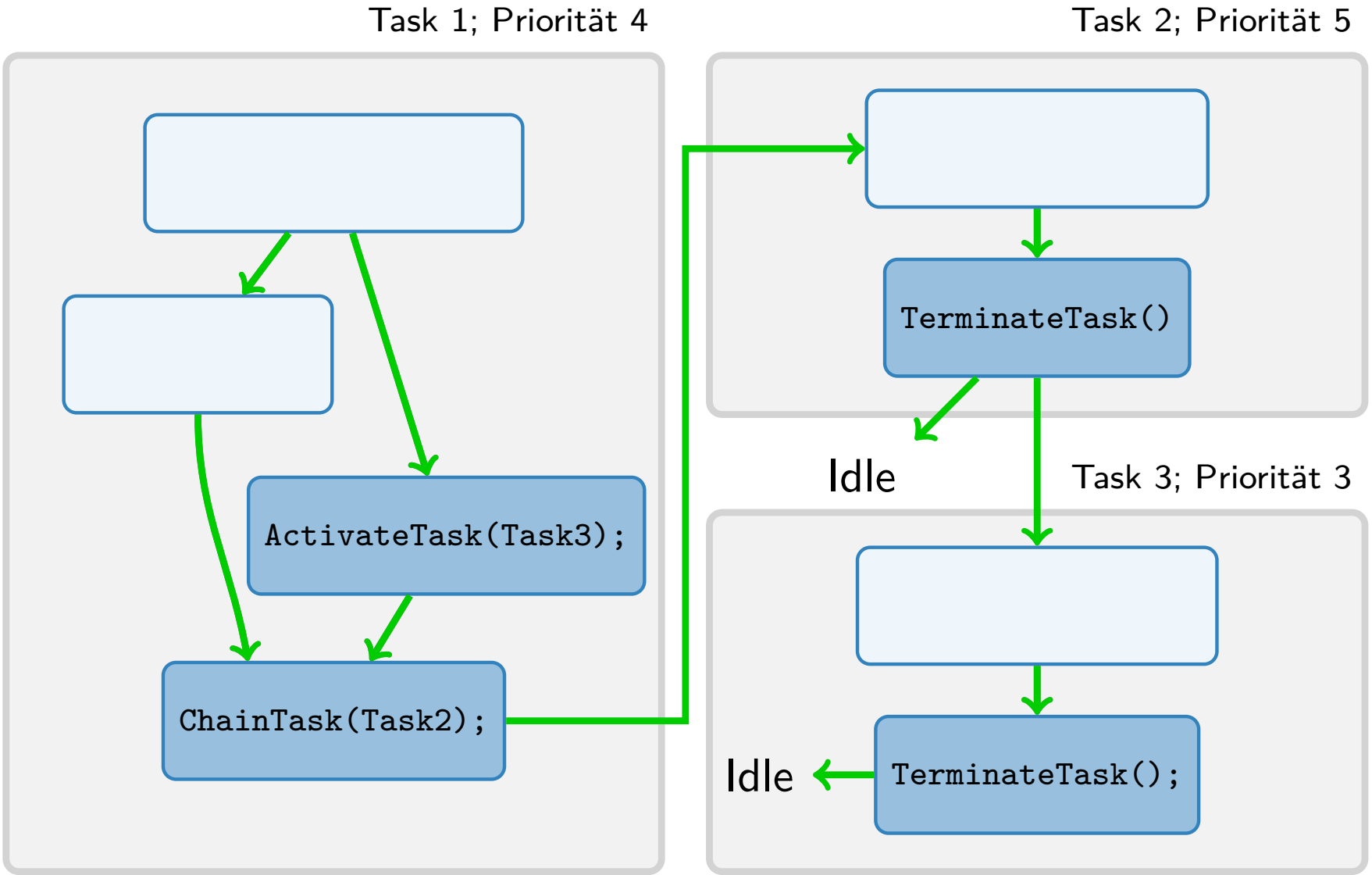
Der Globale Kontrollflussgraph (GCFG)



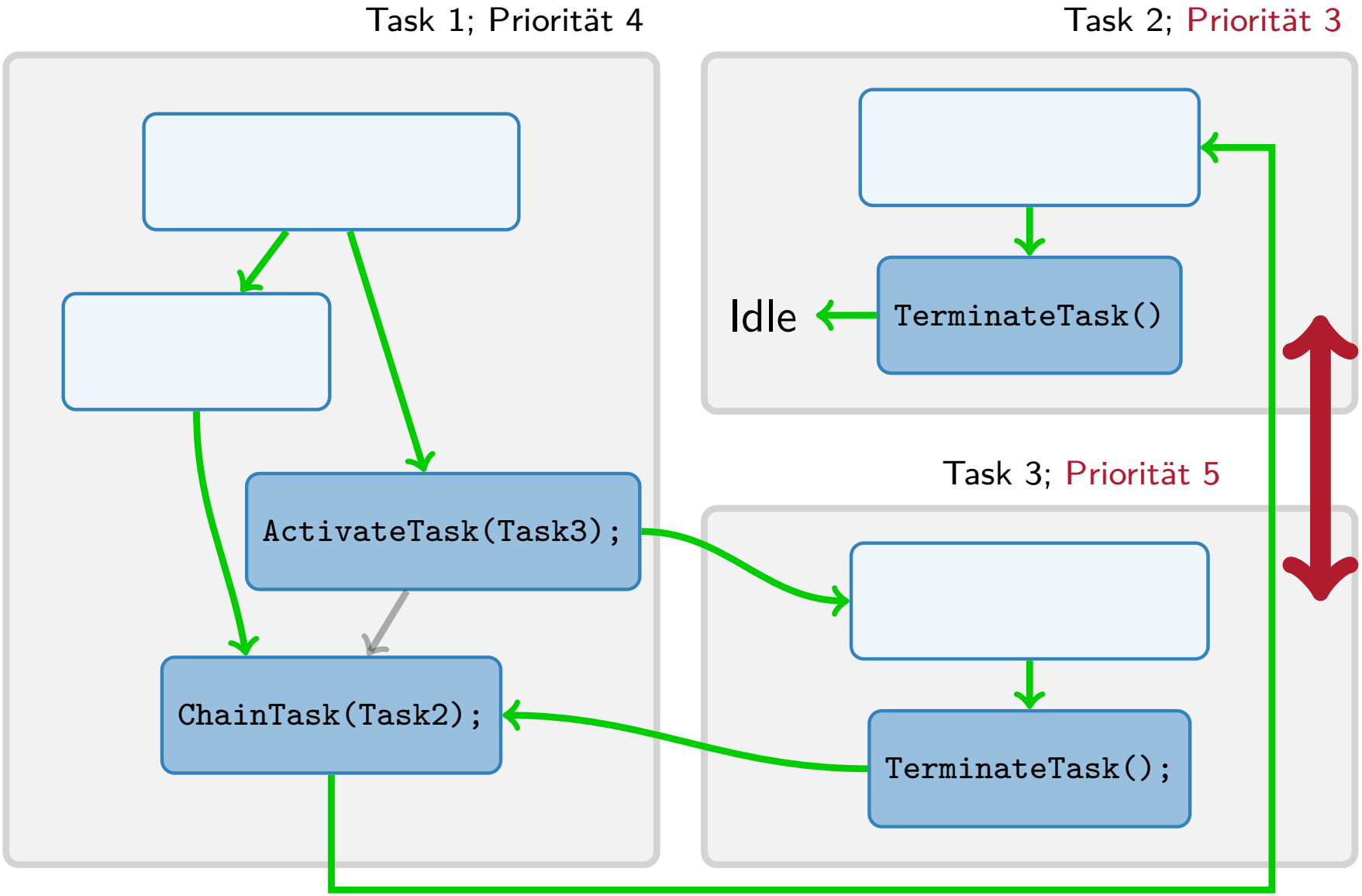
Der Globale Kontrollflussgraph (GCFG)



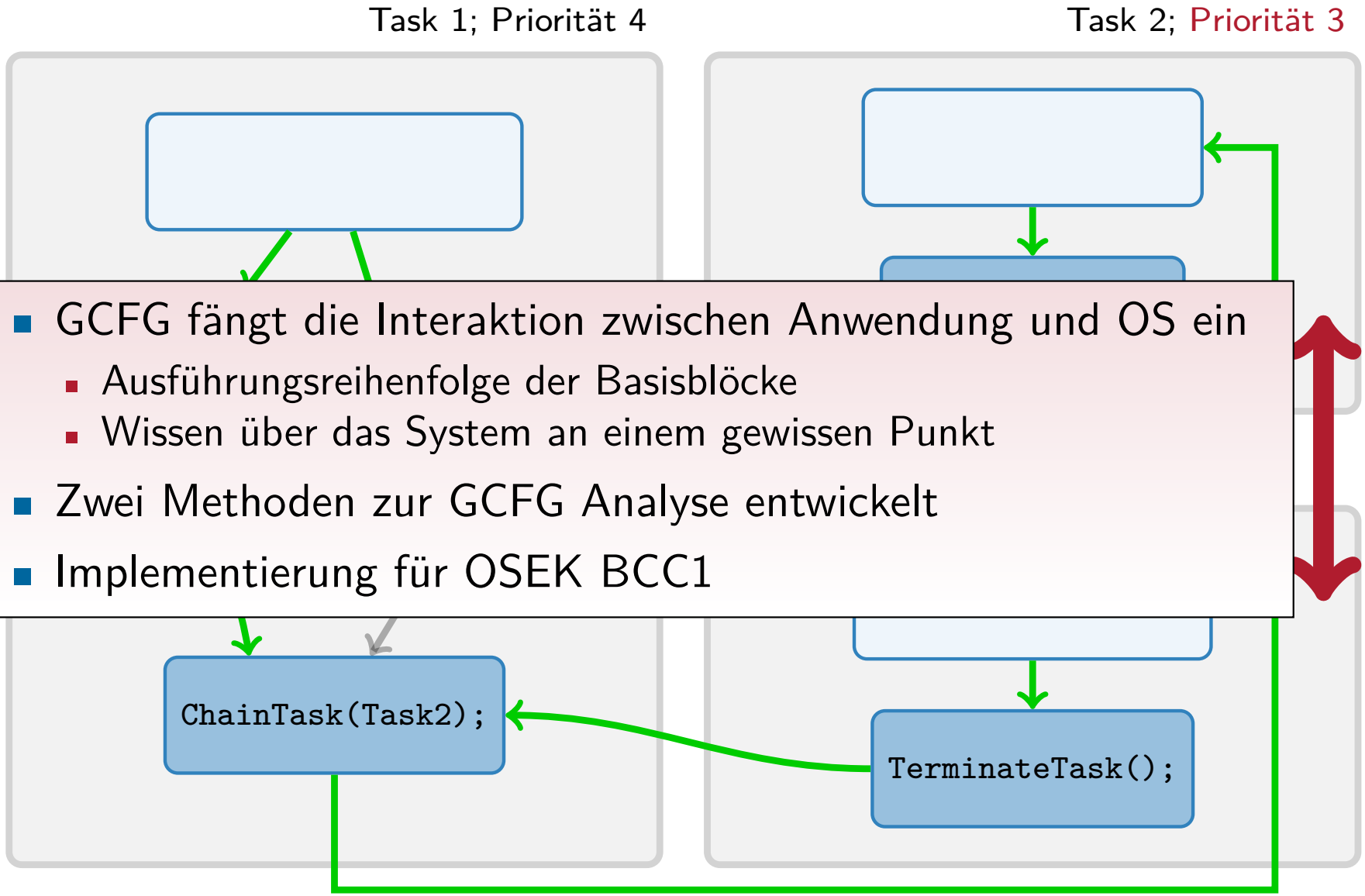
Der Globale Kontrollflussgraph (GCFG)



Ein alternativer globaler Kontrollflussgraph



Ein alternativer globaler Kontrollflussgraph



- GCFG fängt die Interaktion zwischen Anwendung und OS ein
  - Ausführungsreihenfolge der Basisblöcke
  - Wissen über das System an einem gewissen Punkt
- Zwei Methoden zur GCFG Analyse entwickelt
- Implementierung für OSEK BCC1

- **Frage 1:** Wie erhalte ich dieses feingranulares Interaktionswissen?

  ↳ Der globale Kontrollflusgraph

- **Frage 2:** Wie nutze dieses Wissen?

- **Frage 1:** Wie erhalte ich dieses feingranulares Interaktionswissen?
  - ↳ Der globale Kontrollflusgraph

- **Frage 2:** Wie nutze dieses Wissen?
  - ↳ Maßschneiderung der Systemaufrufe

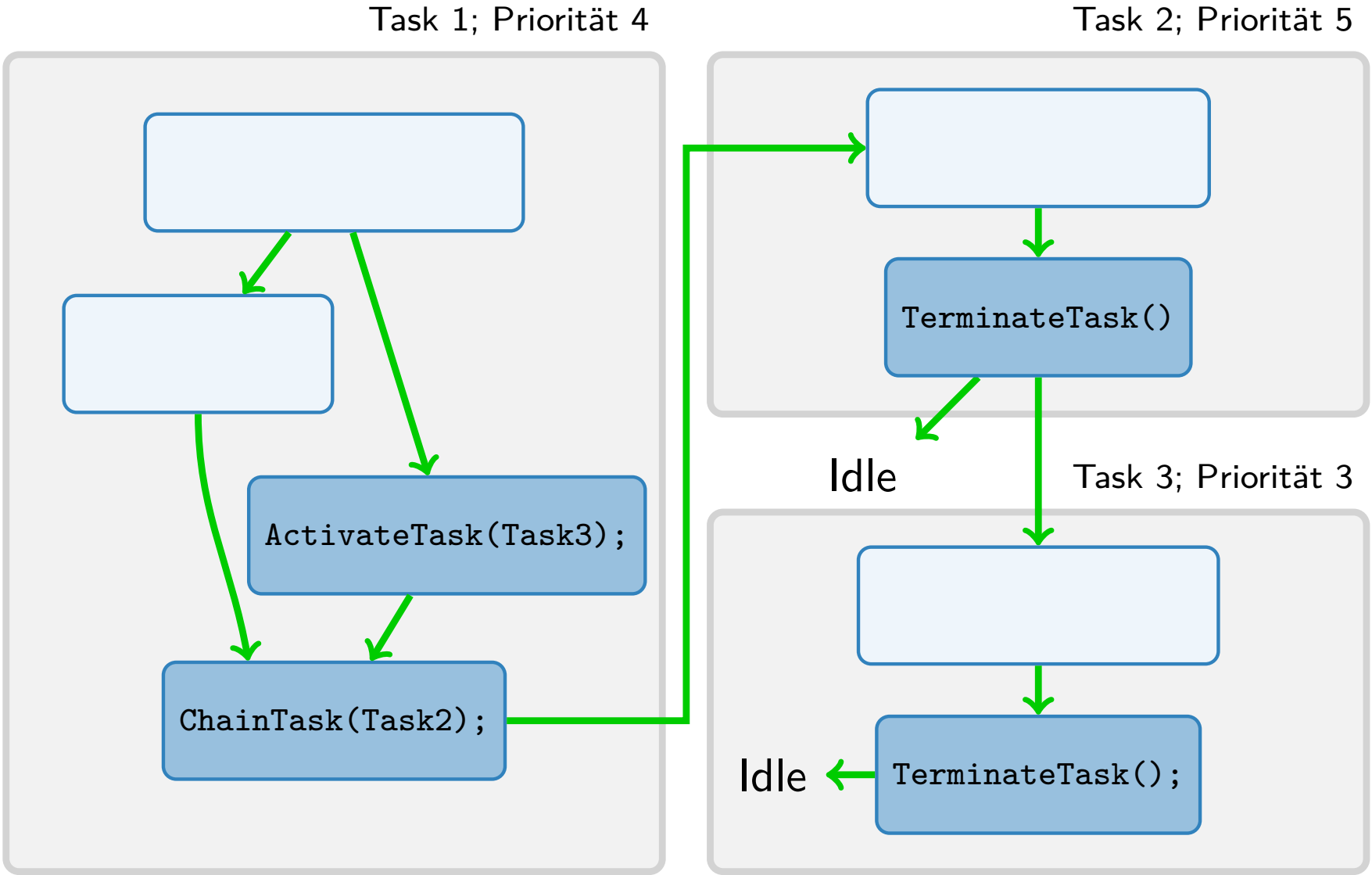
Maßschneiderung des Systems

- Spezialisierung von Systemaufrufen *Laufzeit*
- Zusicherungen über den Zustand des Systems *Fehlertoleranz*
- Kontrollflussüberwachung durch Dominatorregionen *Fehlertoleranz*

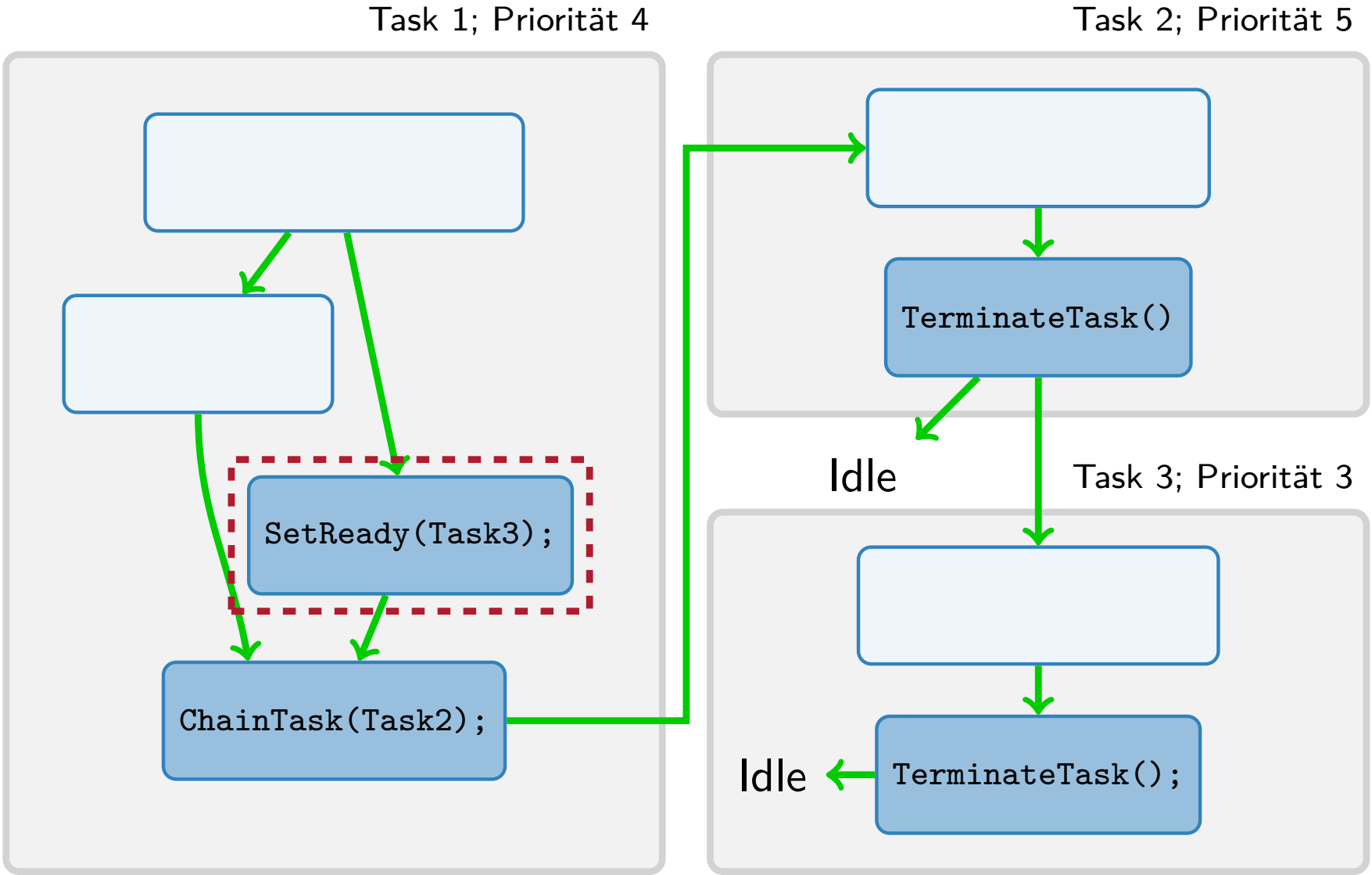
Maßschneiderung des Systems

- Spezialisierung von Systemaufrufen *Laufzeit*
  - Erzeugung spezialisierter Kernelfragmente
  - Information des GCFG beschreibt mögliche Einplanungsentscheidungen

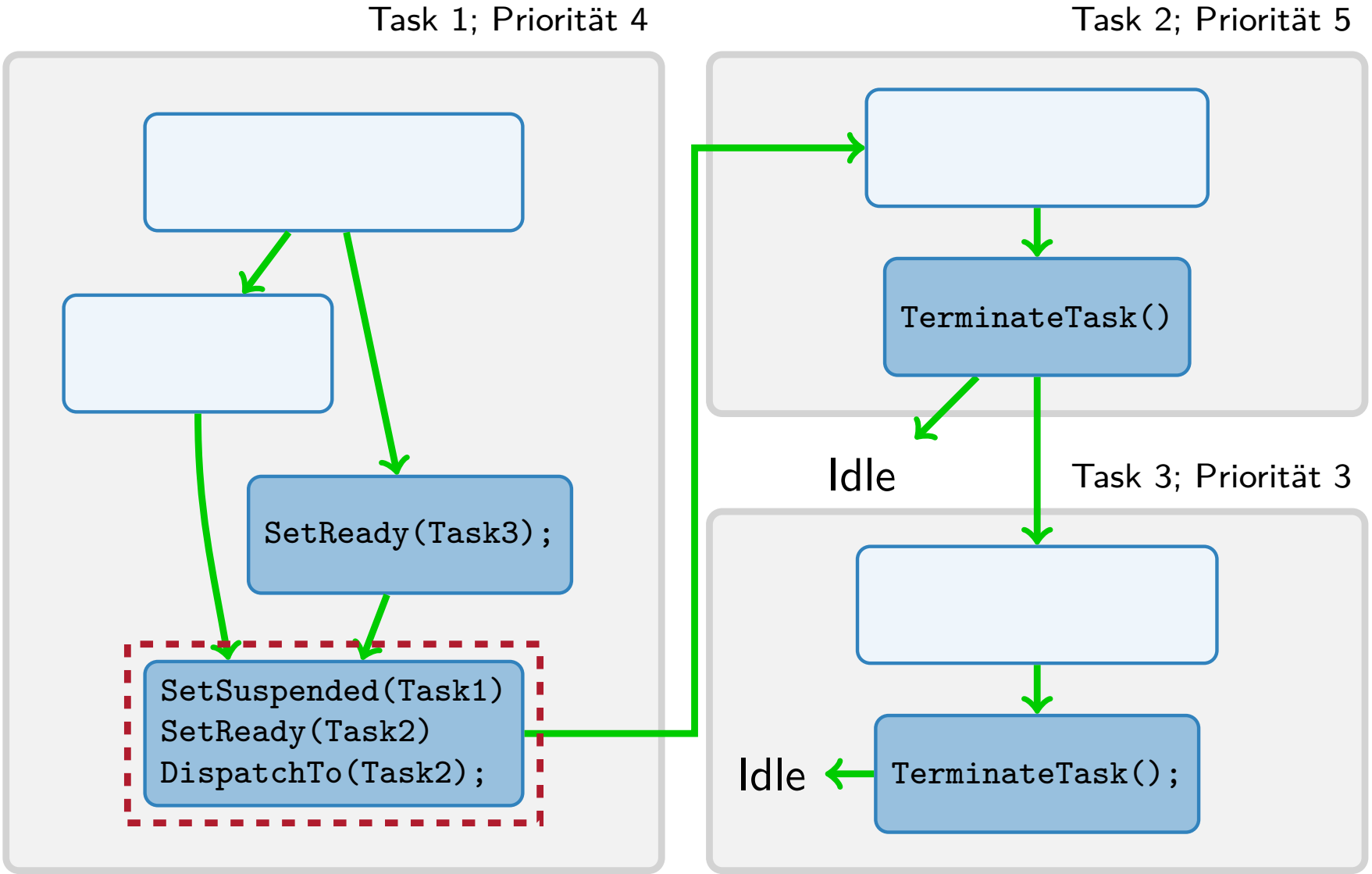
Spezialisierung von Systemaufrufen



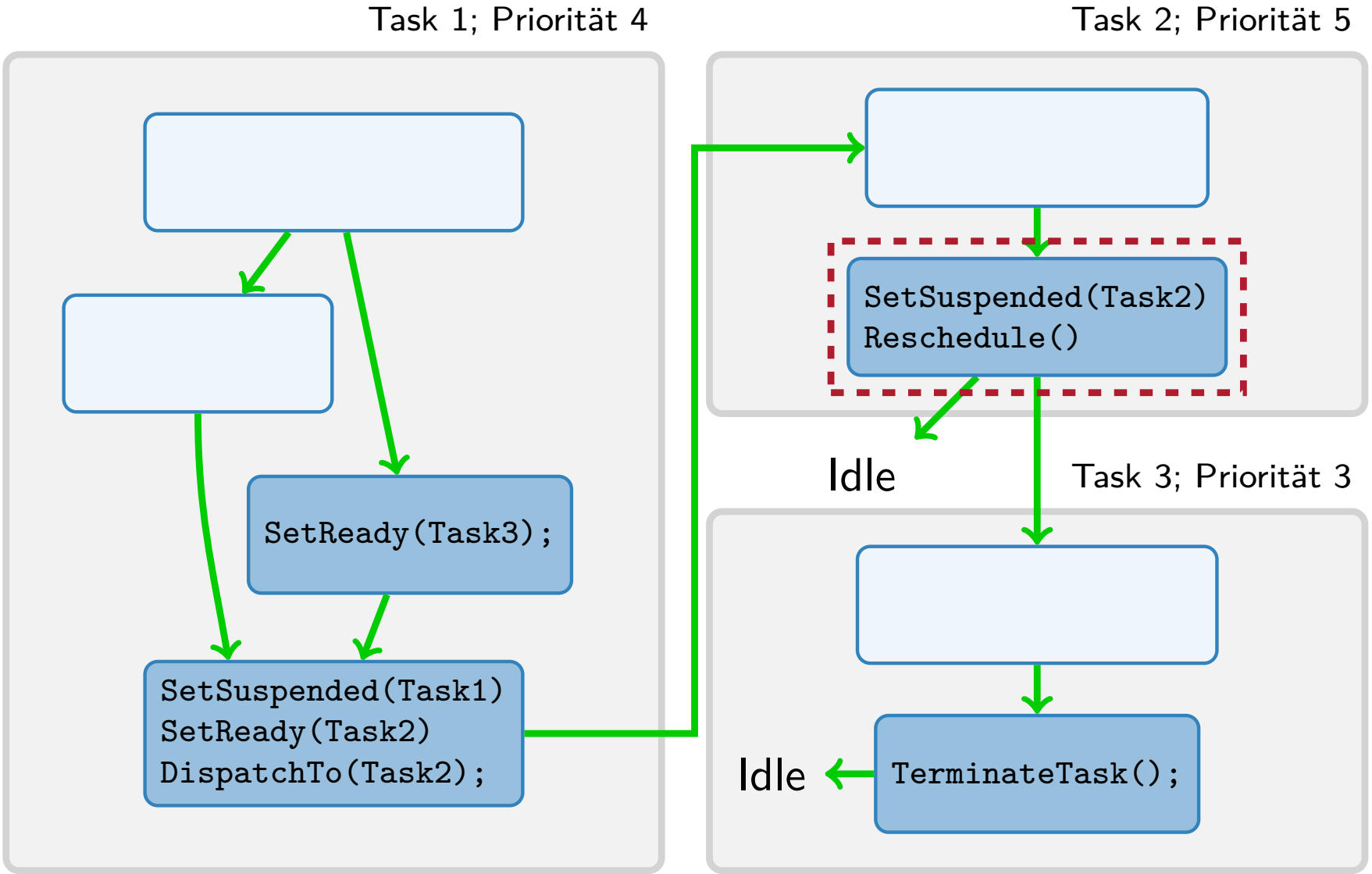
Spezialisierung von Systemaufrufen



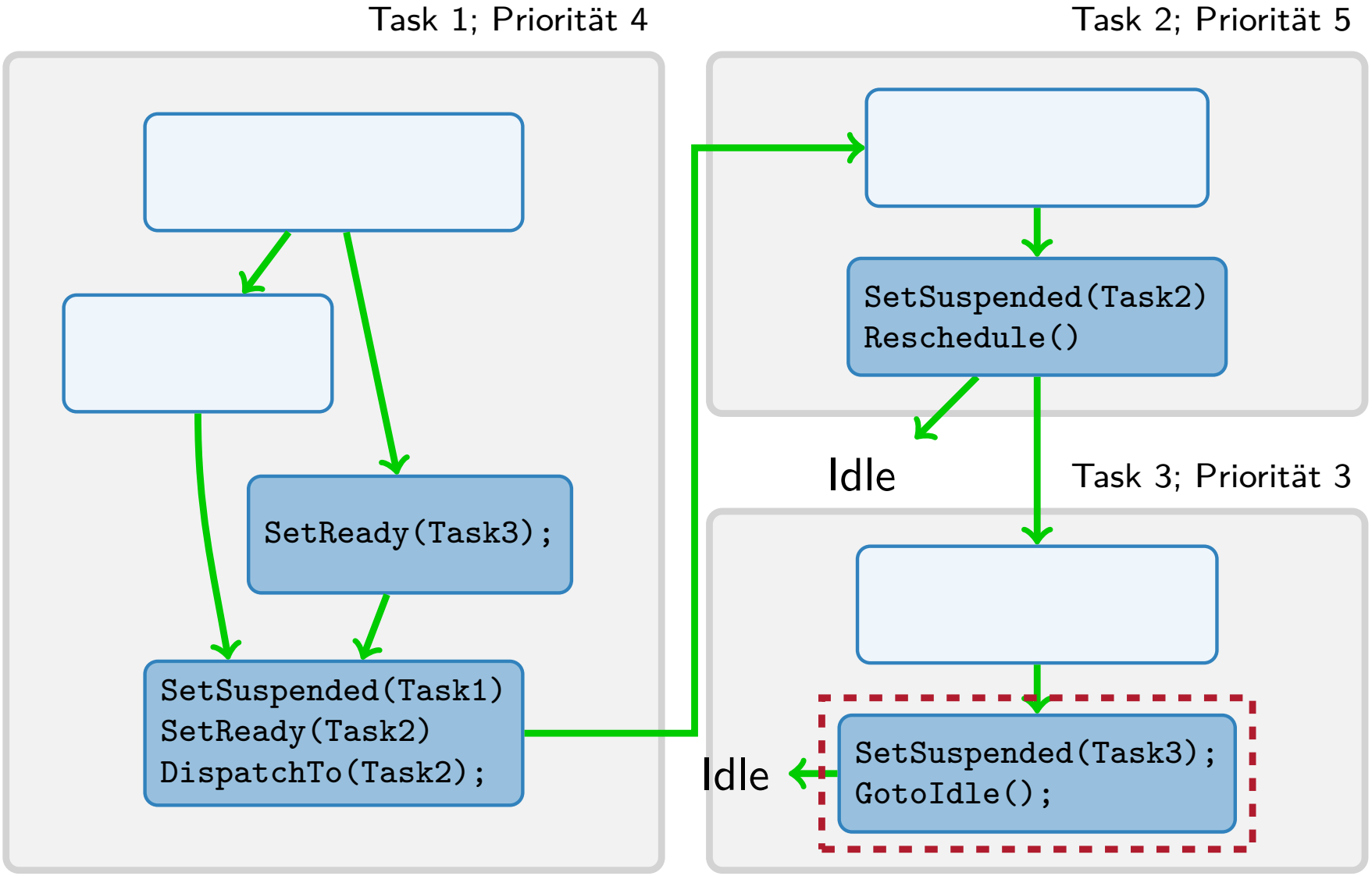
Spezialisierung von Systemaufrufen



Spezialisierung von Systemaufrufen



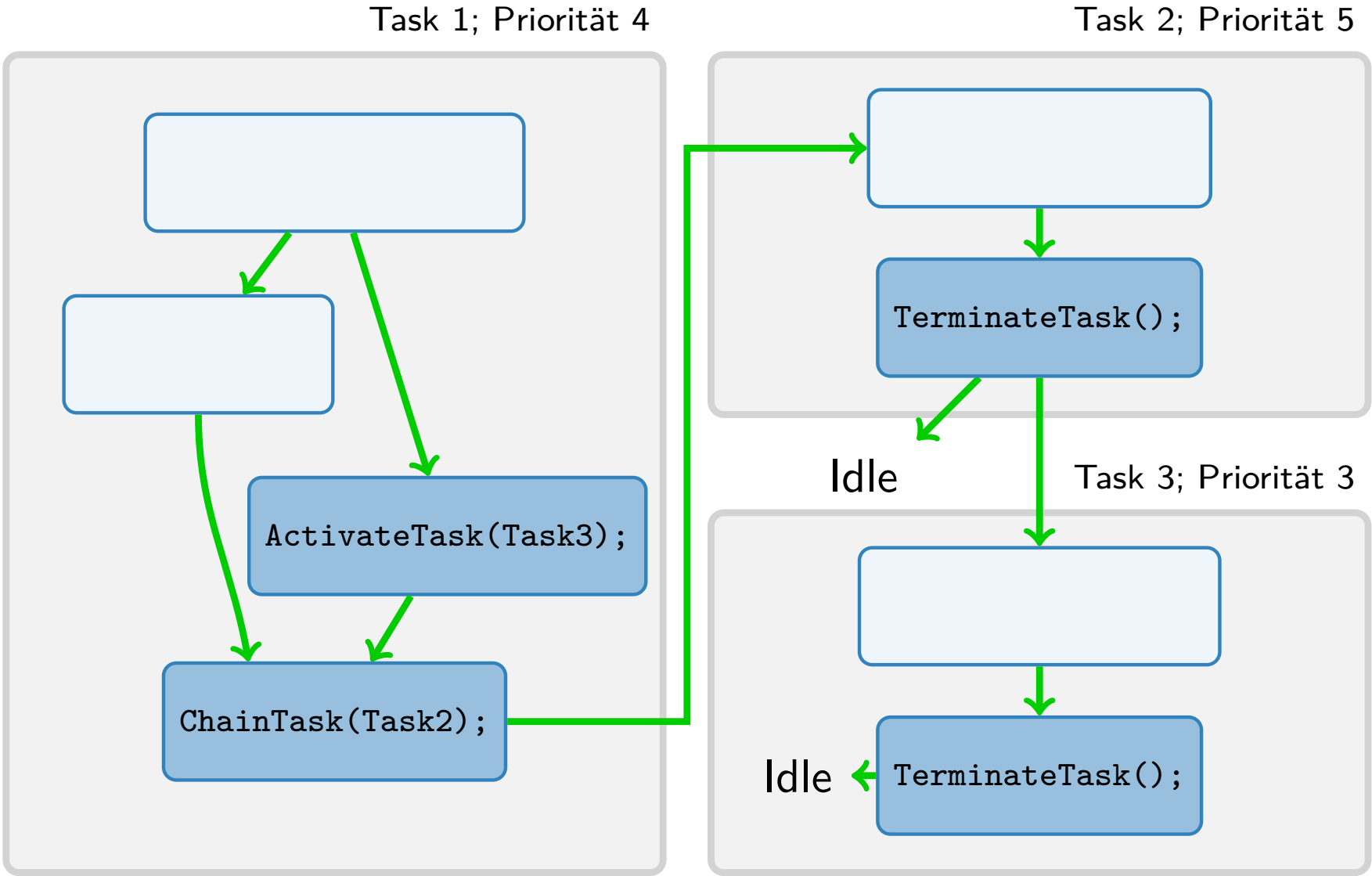
Spezialisierung von Systemaufrufen



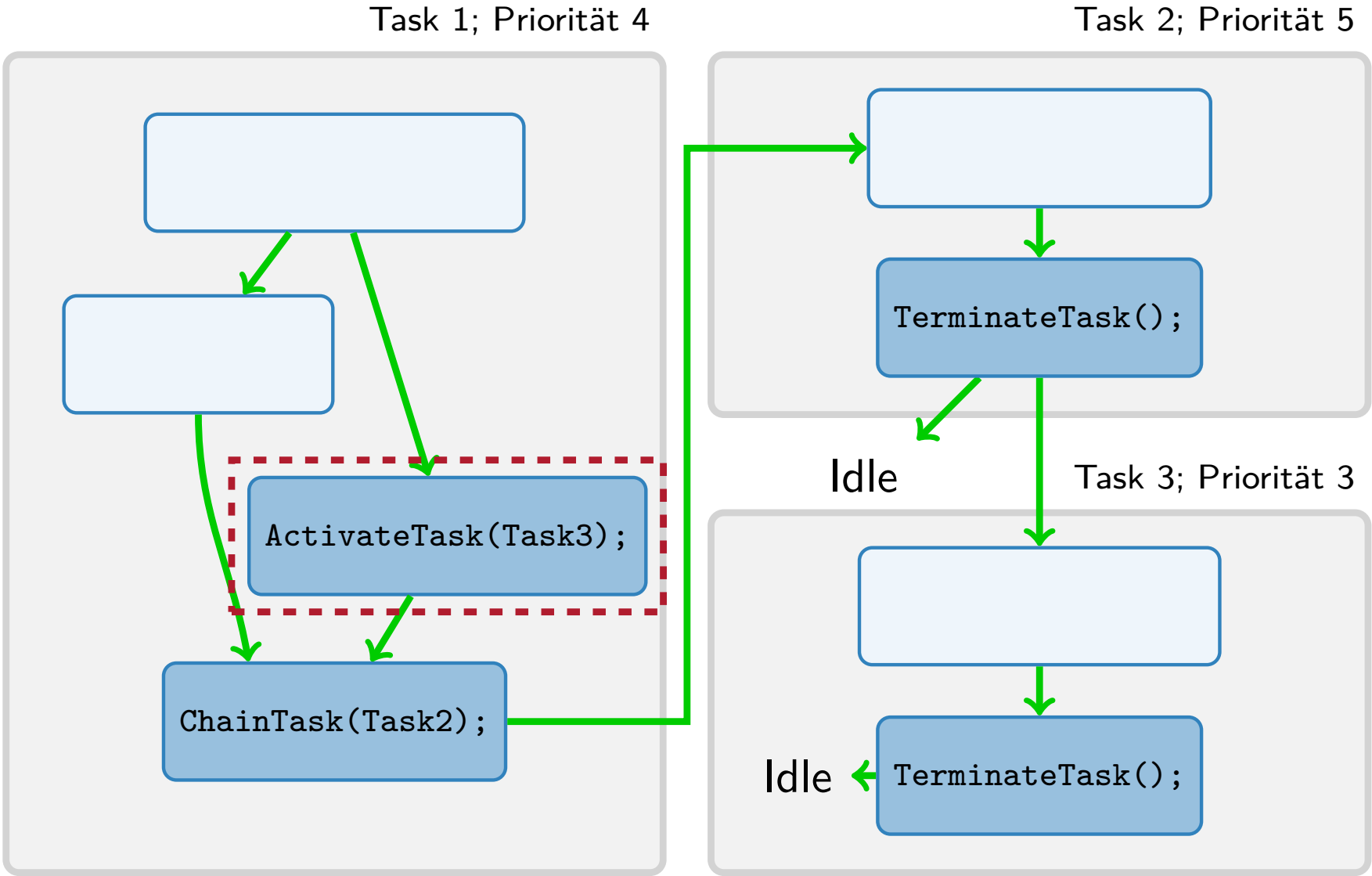
Maßschneiderung des Systems

- Spezialisierung von Systemaufrufen *Laufzeit*
- Zusicherungen über den Zustand des Systems *Fehlertoleranz*
  - Für jeden Systemaufruf stellt der GCFG Zustandsinformationen bereit
  - Wir fügen Zusicherungen auf konstante Felder ein

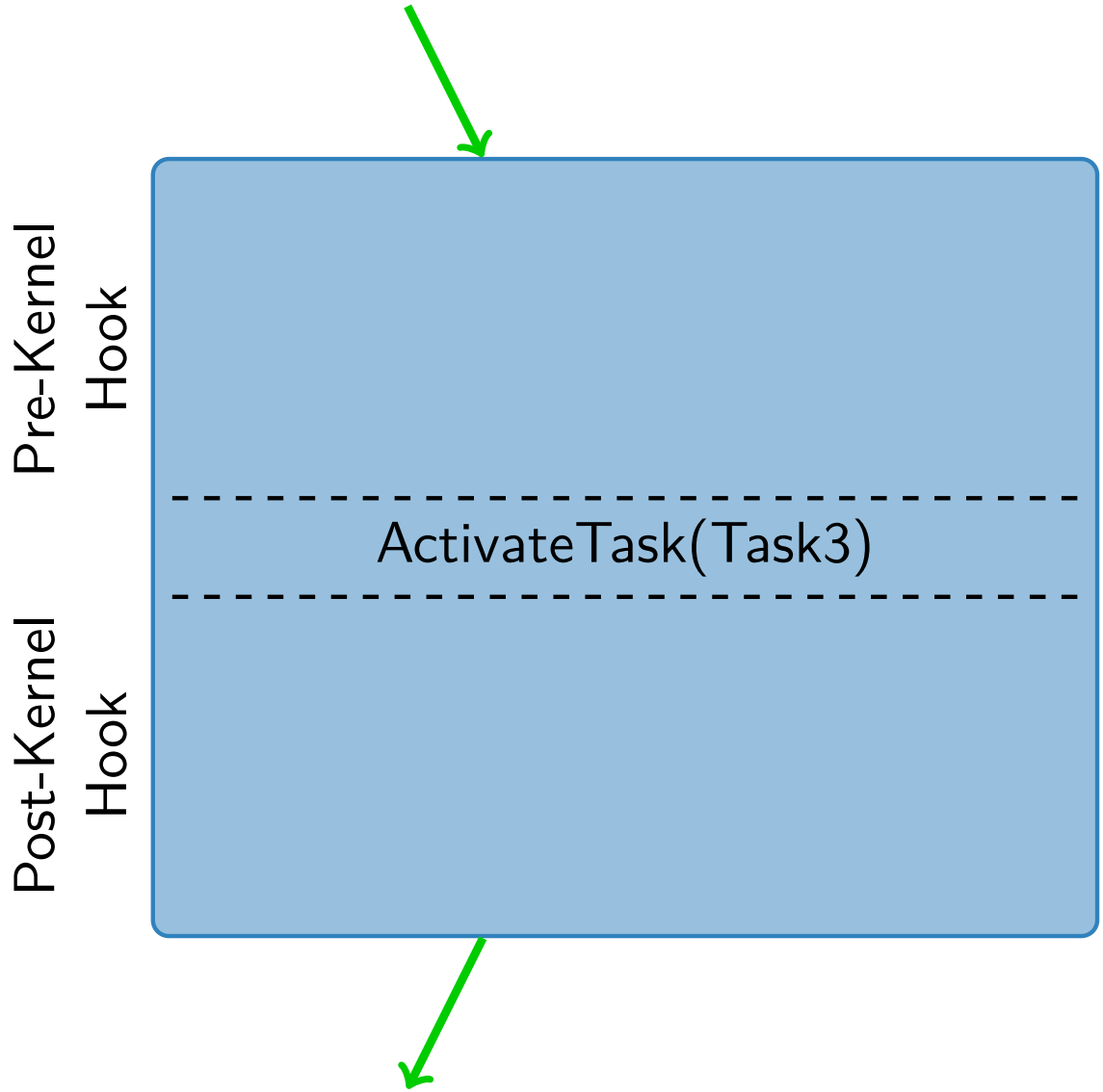
Zusicherungen über den Systemzustand



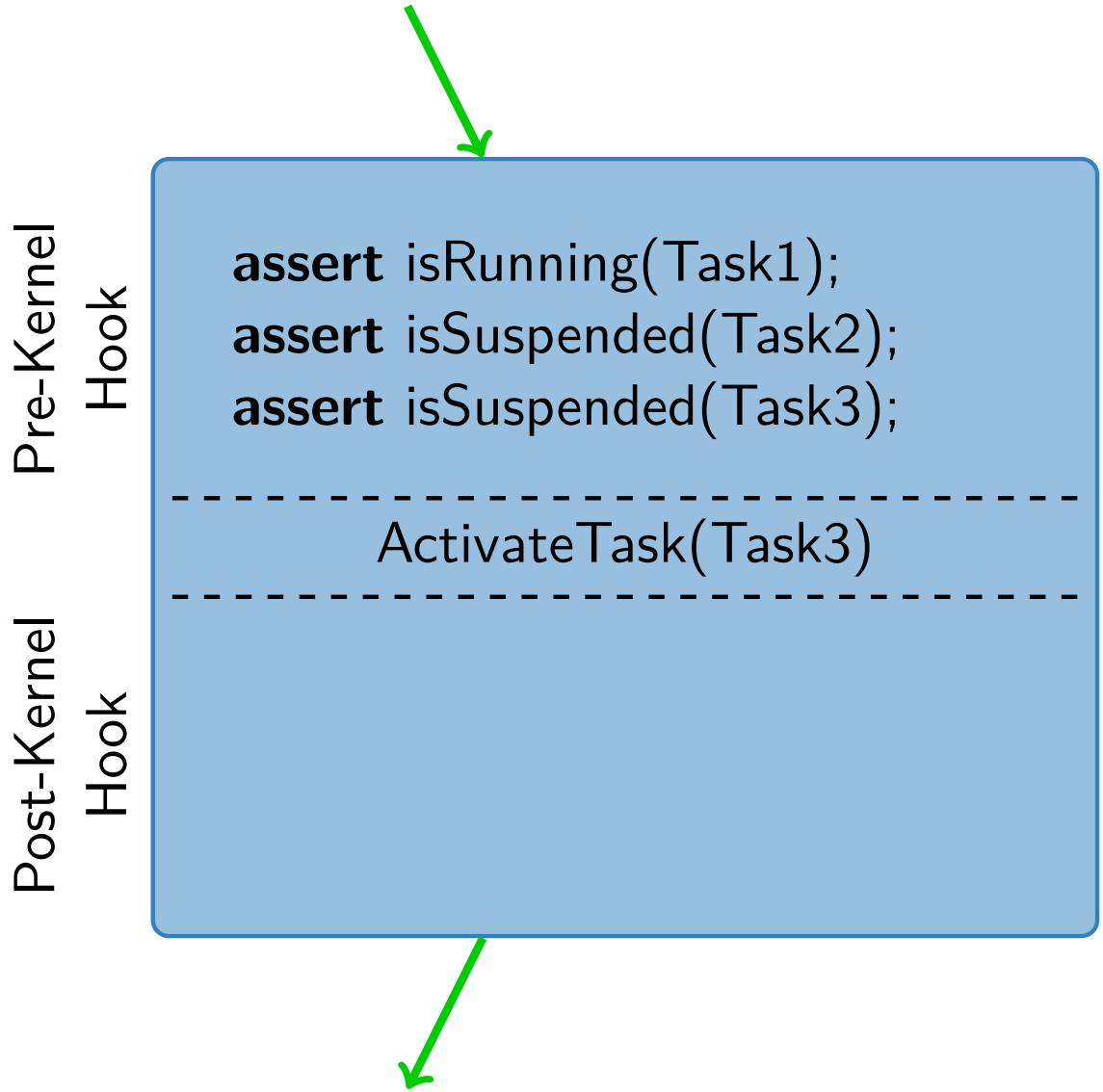
Zusicherungen über den Systemzustand



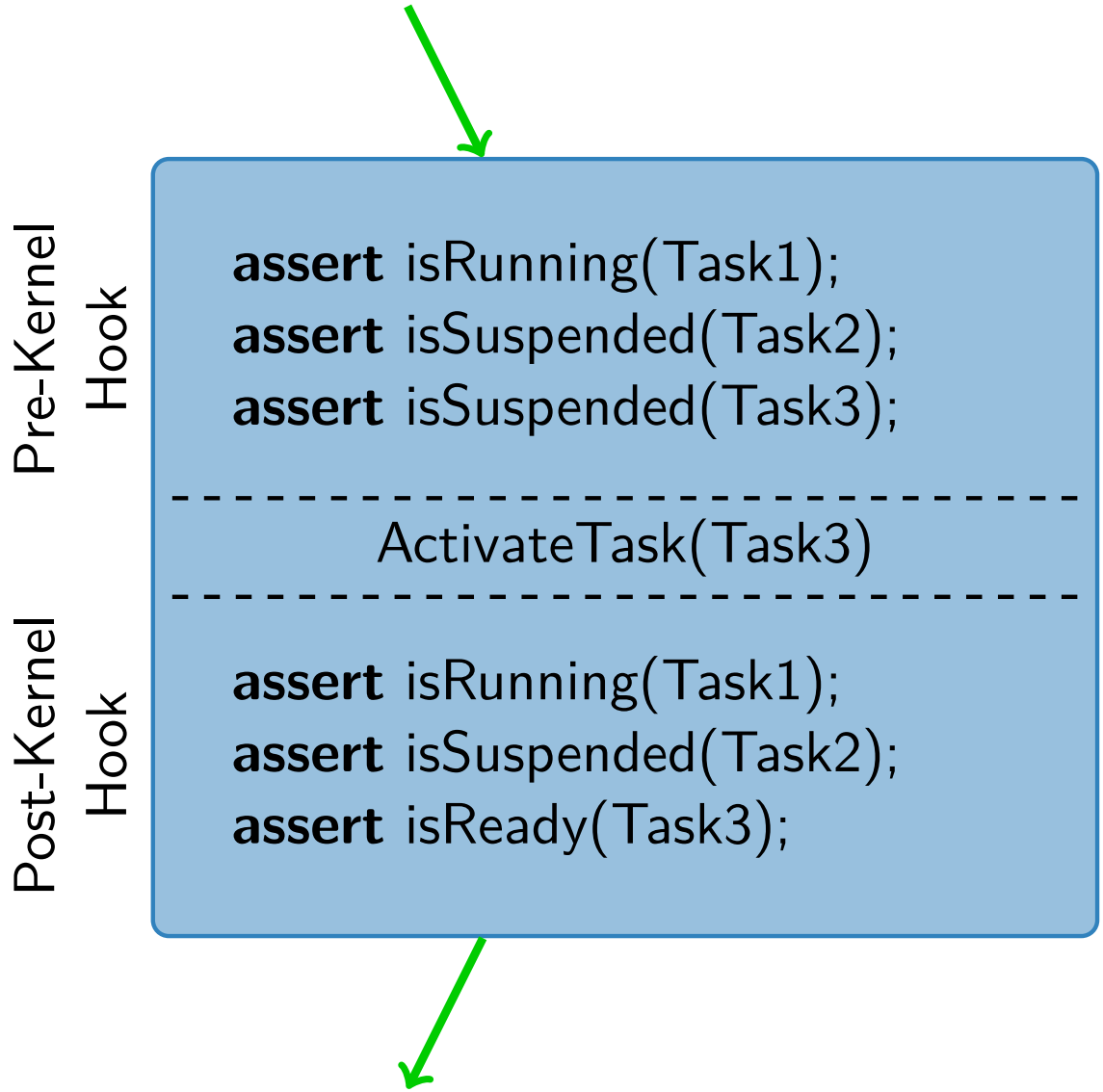
Zusicherungen über den Systemzustand



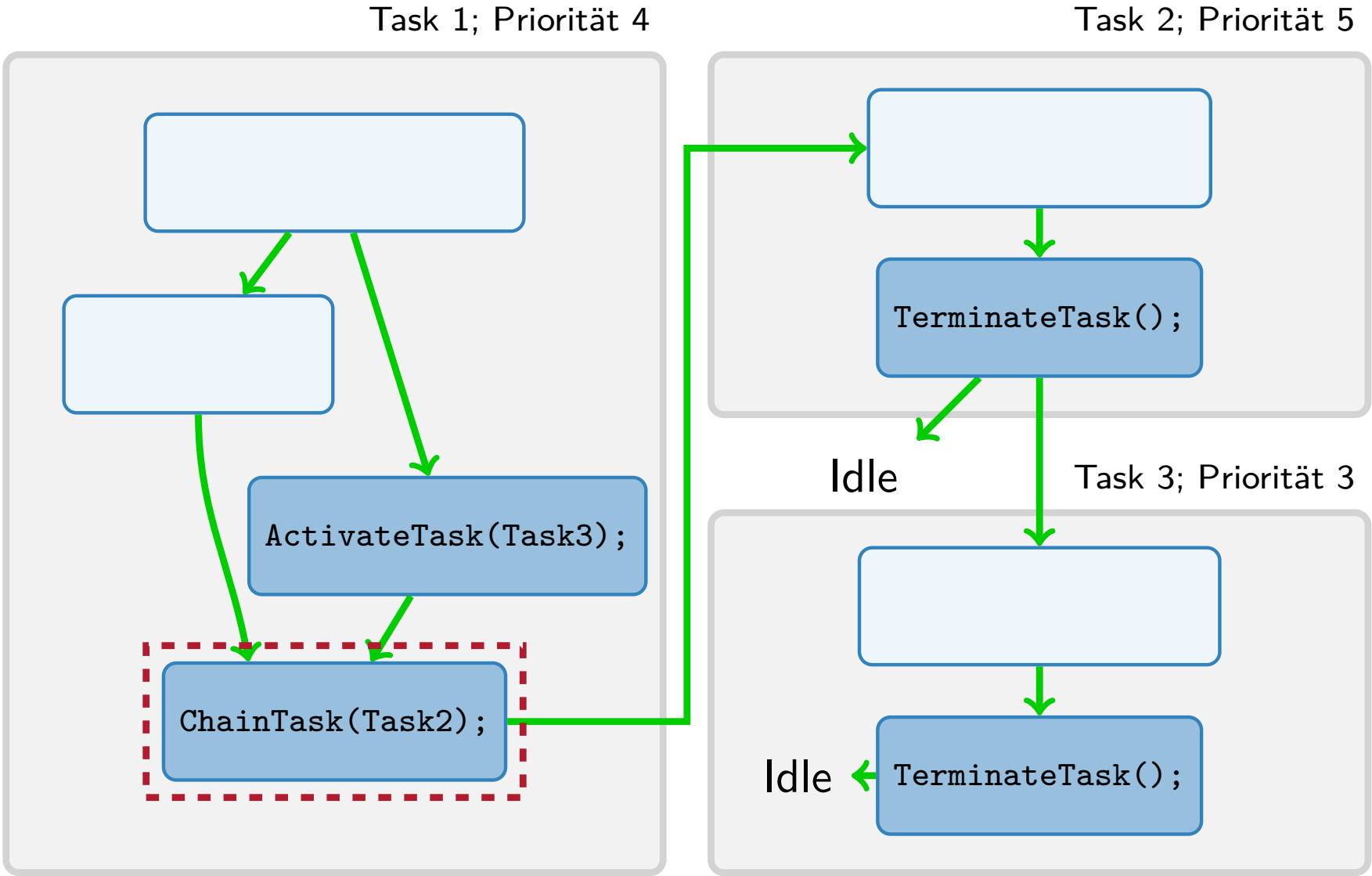
Zusicherungen über den Systemzustand



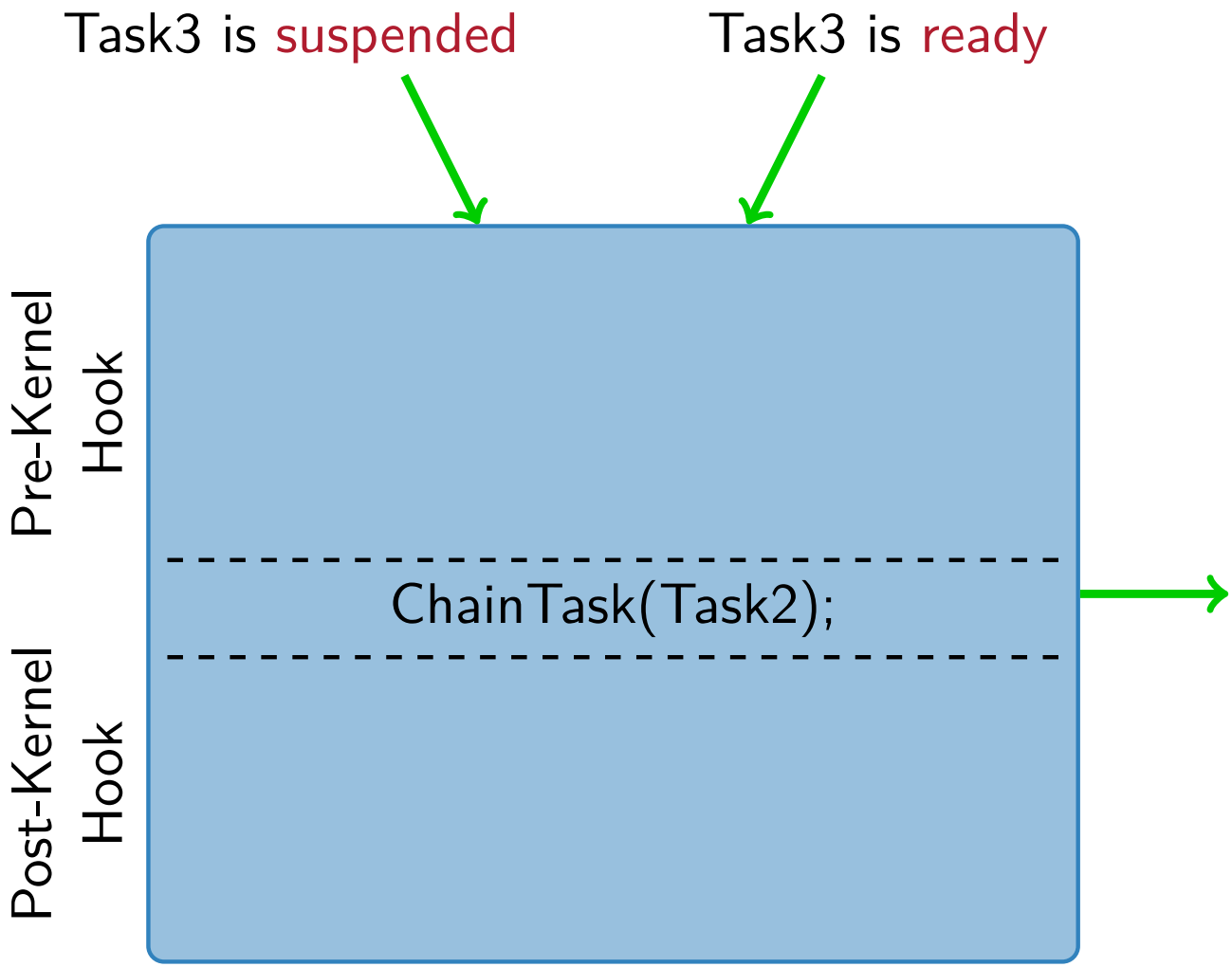
Zusicherungen über den Systemzustand



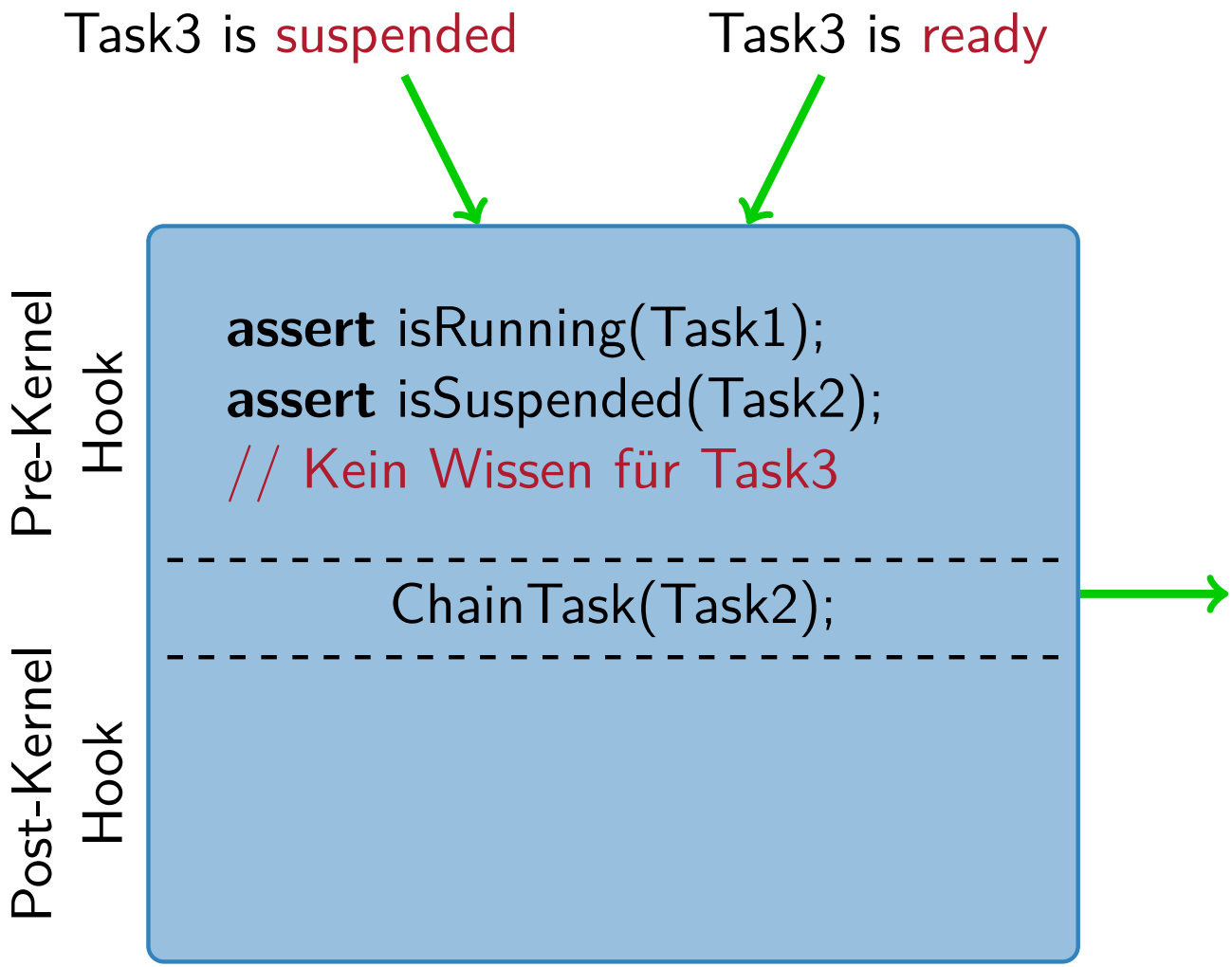
Zusicherungen über den Systemzustand



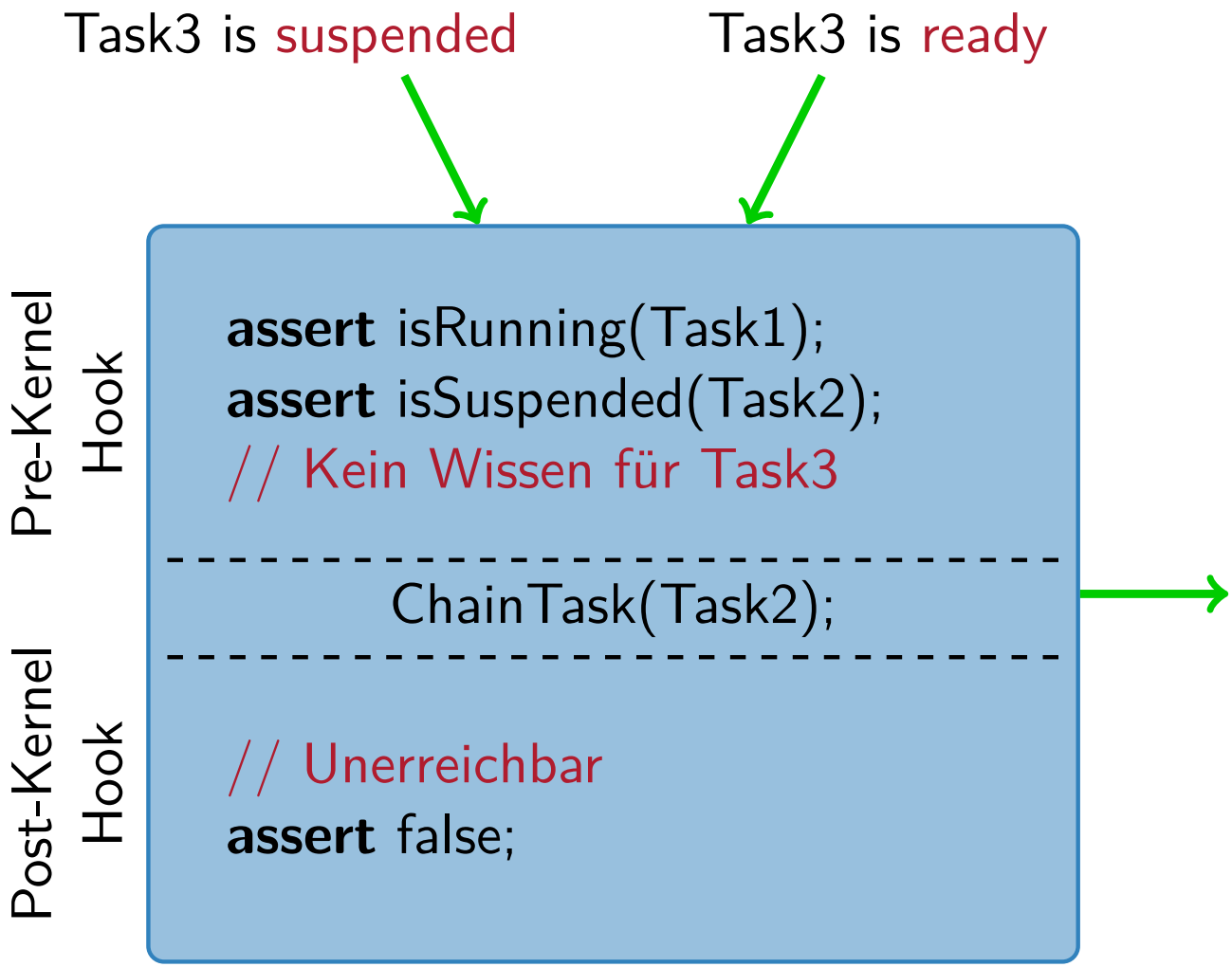
Zusicherungen über den Systemzustand



Zusicherungen über den Systemzustand



Zusicherungen über den Systemzustand



- **Frage 1:** Wie erhalte ich dieses feingranulares Interaktionswissen?
  - ↳ Der globale Kontrollflusgraph

- **Frage 2:** Wie nutze dieses Wissen?
  - ↳ Maßschneiderung der Systemaufrufe

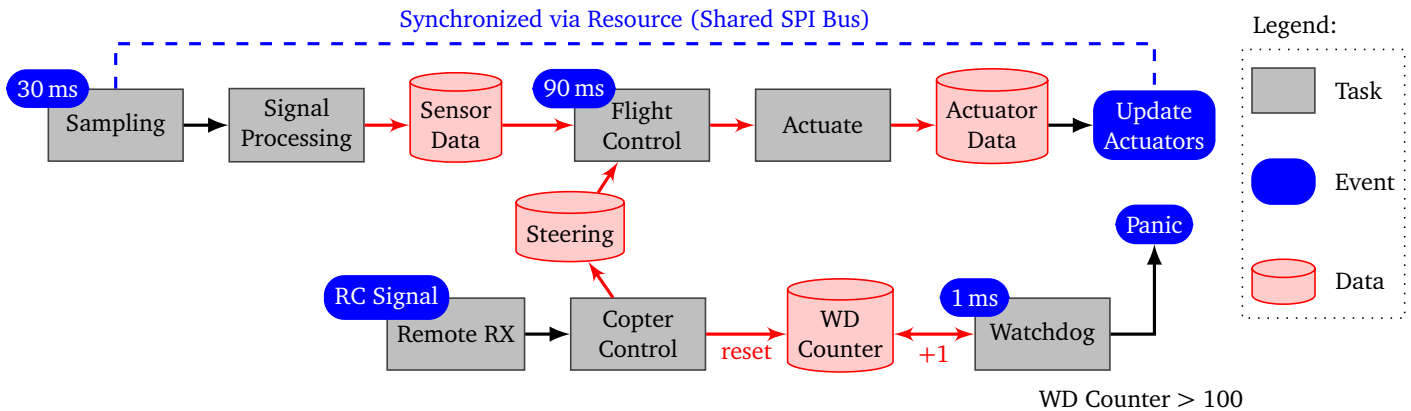
Auswertung der Maßschneiderung

Evaluationszenario

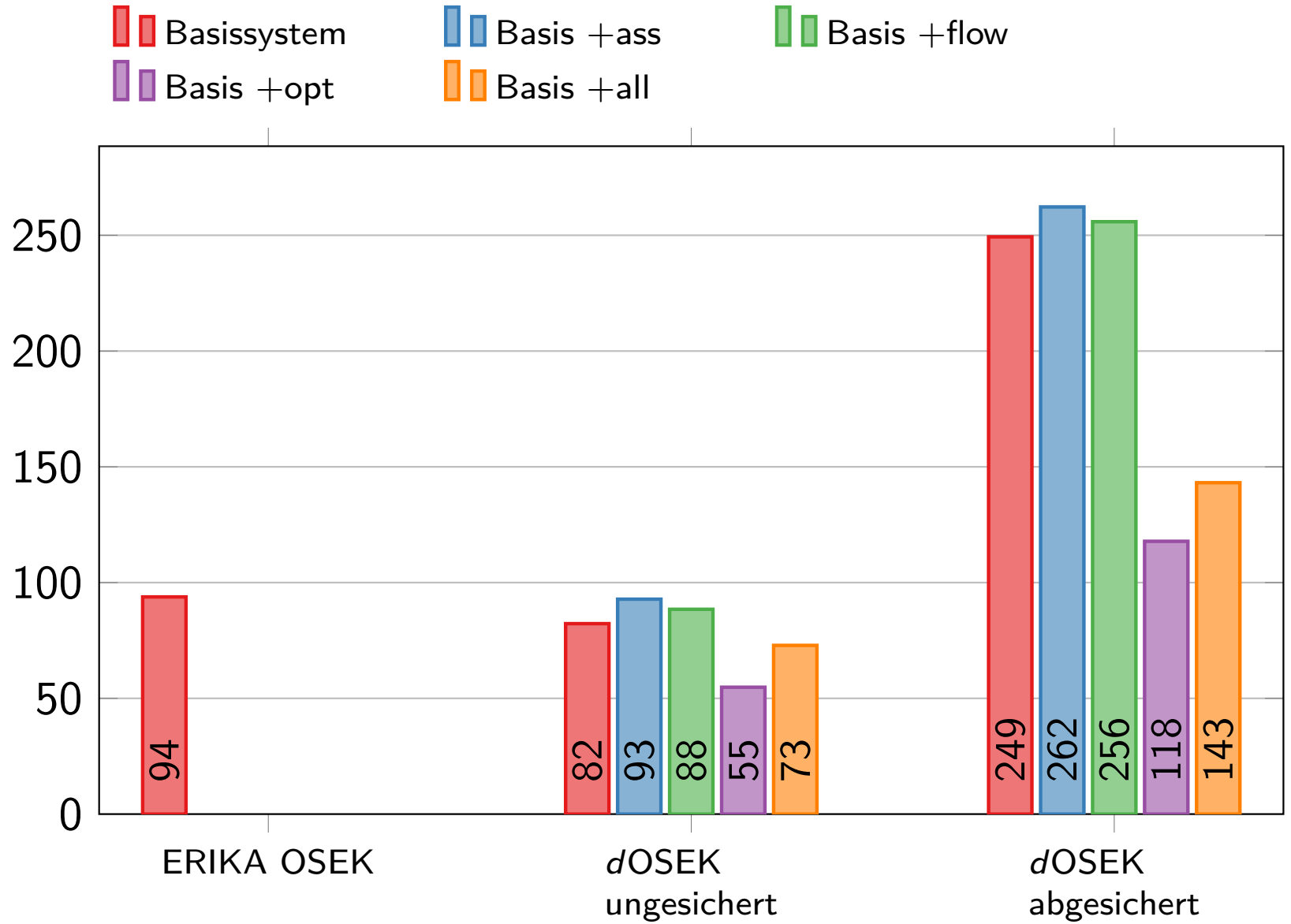
- Evaluationsystem: *d*OSEK (*dependable* OSEK)
  - Fehlertolerante OSEK Implementierung für IA-32
  - Große Laufzeit durch arithmetische Codierung (+500 %)
  - Generativer Ansatz

Evaluationszenario

- Evaluationsystem: *d*OSEK (*dependable* OSEK)
  - Fehlertolerante OSEK Implementierung für IA-32
  - Große Laufzeit durch arithmetische Codierung (+500 %)
  - Generativer Ansatz
- Evaluationszenario: *I4Copter*
  - Anwendungscode durch Prüfpunkte ersetzt
  - 11 Tasks, 3 Alarme, 1 Unterbrecher
  - Ausführung von 3 Hyperperioden (172 Prüfpunkte)



Laufzeit durch den Kernel



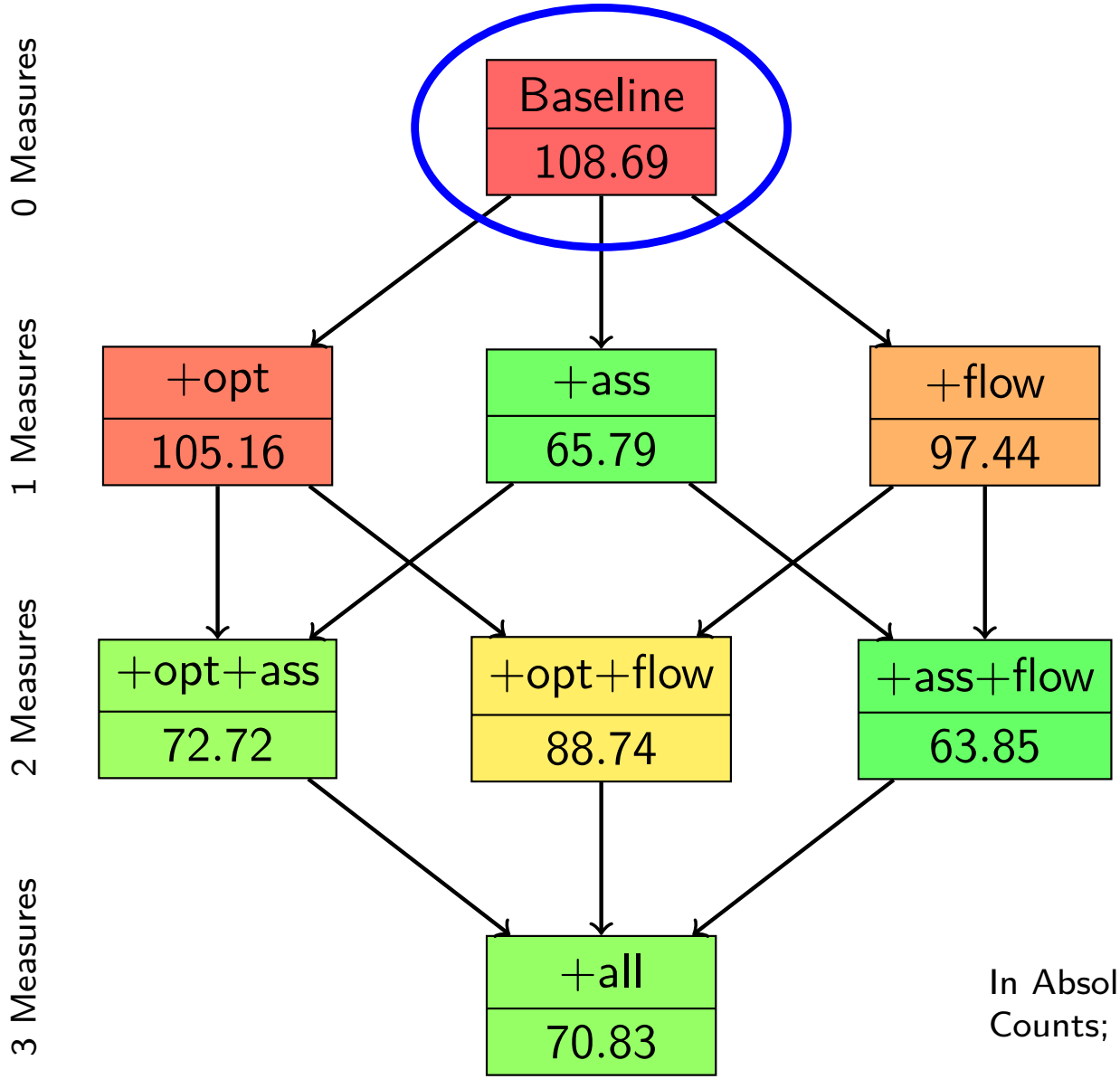
Speicher- und Registerfehler: FAIL*

- FAIL* Fehlerinjektionswerkzeug
  - Fehlerinjektion in einen, deterministischen Systemablauf
  - Simulation des gesamten Systems in virtueller Maschine (Bochs; IA-32)

Speicher- und Registerfehler: FAIL*

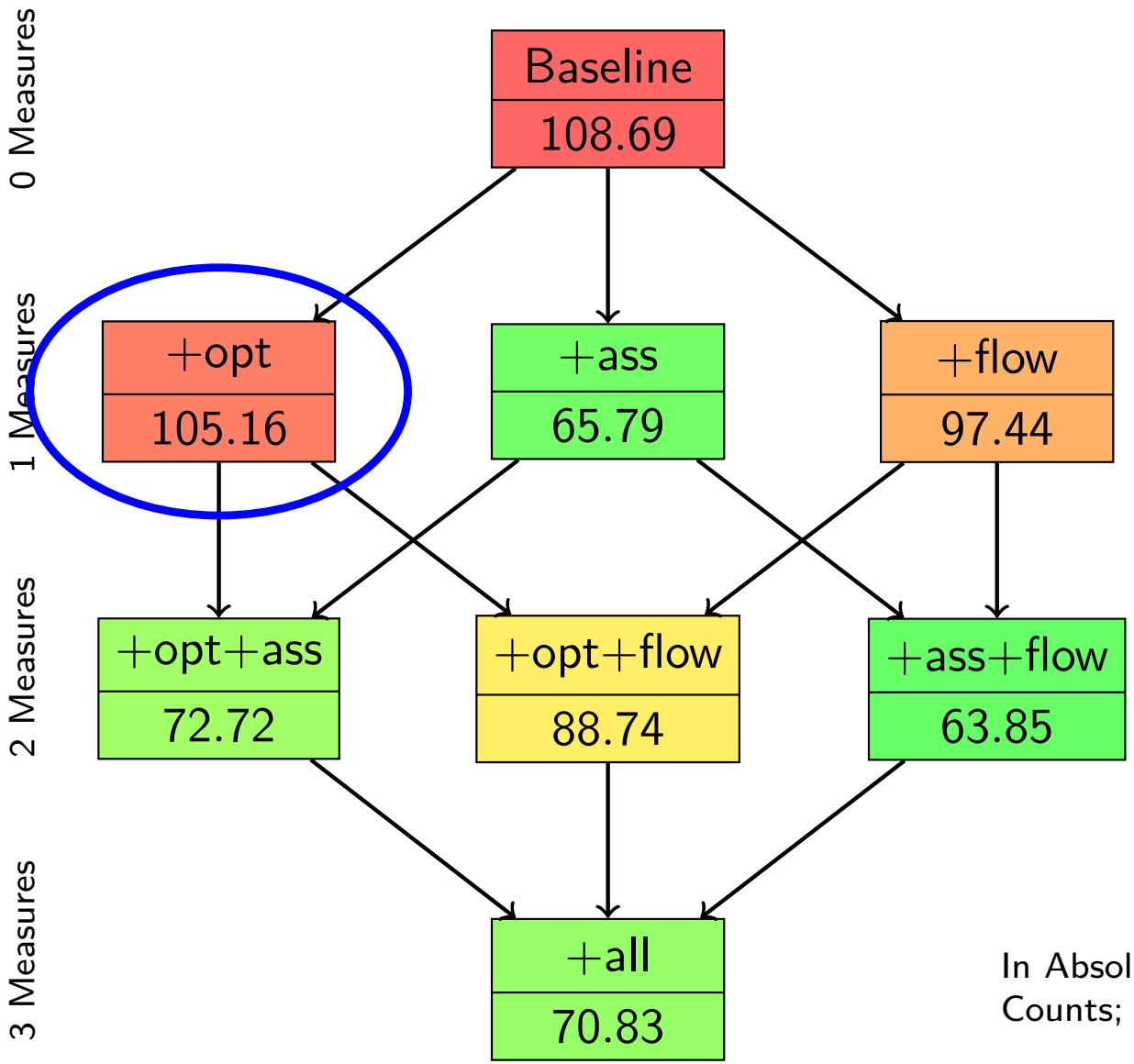
- FAIL* Fehlerinjektionswerkzeug
  - Fehlerinjektion in einen, deterministischen Systemablauf
  - Simulation des gesamten Systems in virtueller Maschine (Bochs; IA-32)
- Ziel: Senkung der Silent Data Corruptions (SDCs)
  - SDC := Das System verhält sich fehlerhaft ohne dies zu bemerken
  - Fehlerhafter Ablauf der 172 Prüfpunkte ist eine SDC
  - Beschädigung der Anwendungsdaten (Stack) ist eine SDC

Fehlerinjektionsergebnisse – SDCs



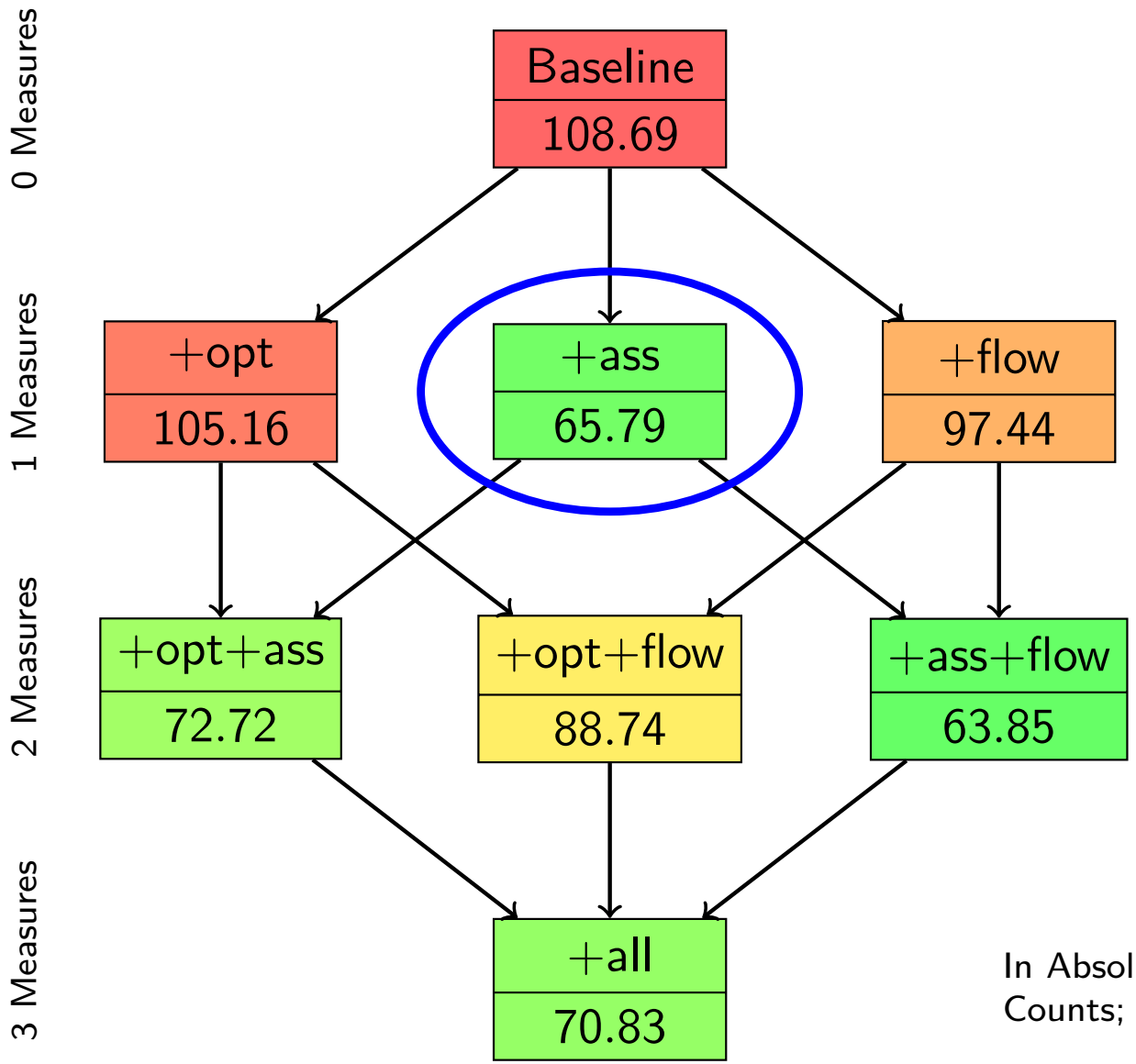
In Absolute Fault Counts; Factor $\cdot 10^3$

Fehlerinjektionsergebnisse – SDCs



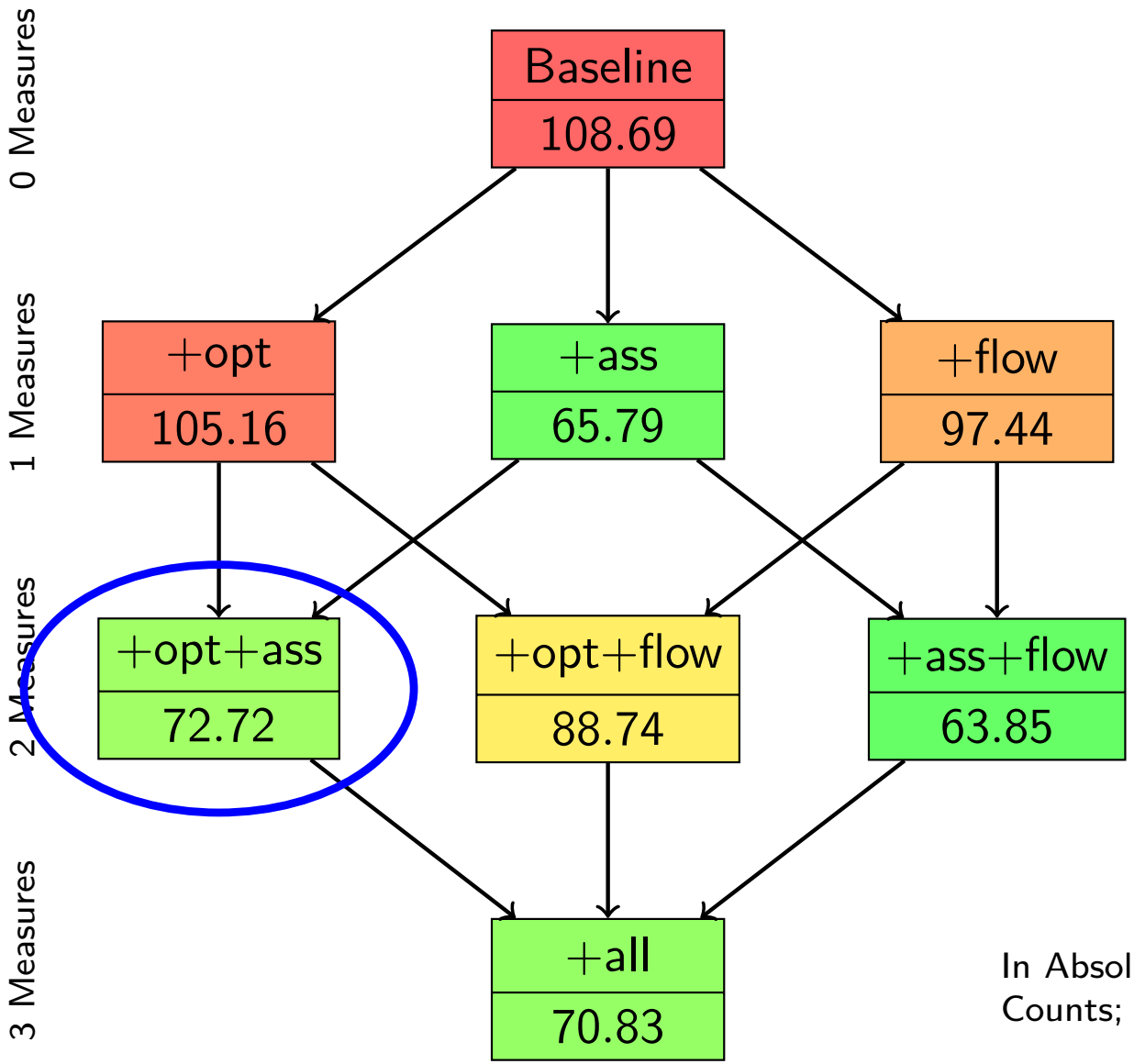
In Absolute Fault Counts; Factor $\cdot 10^3$

Fehlerinjektionsergebnisse – SDCs



In Absolute Fault Counts; Factor $\cdot 10^3$

Fehlerinjektionsergebnisse – SDCs



In Absolute Fault Counts; Factor $\cdot 10^3$

Zusammenfassung und Ausblick

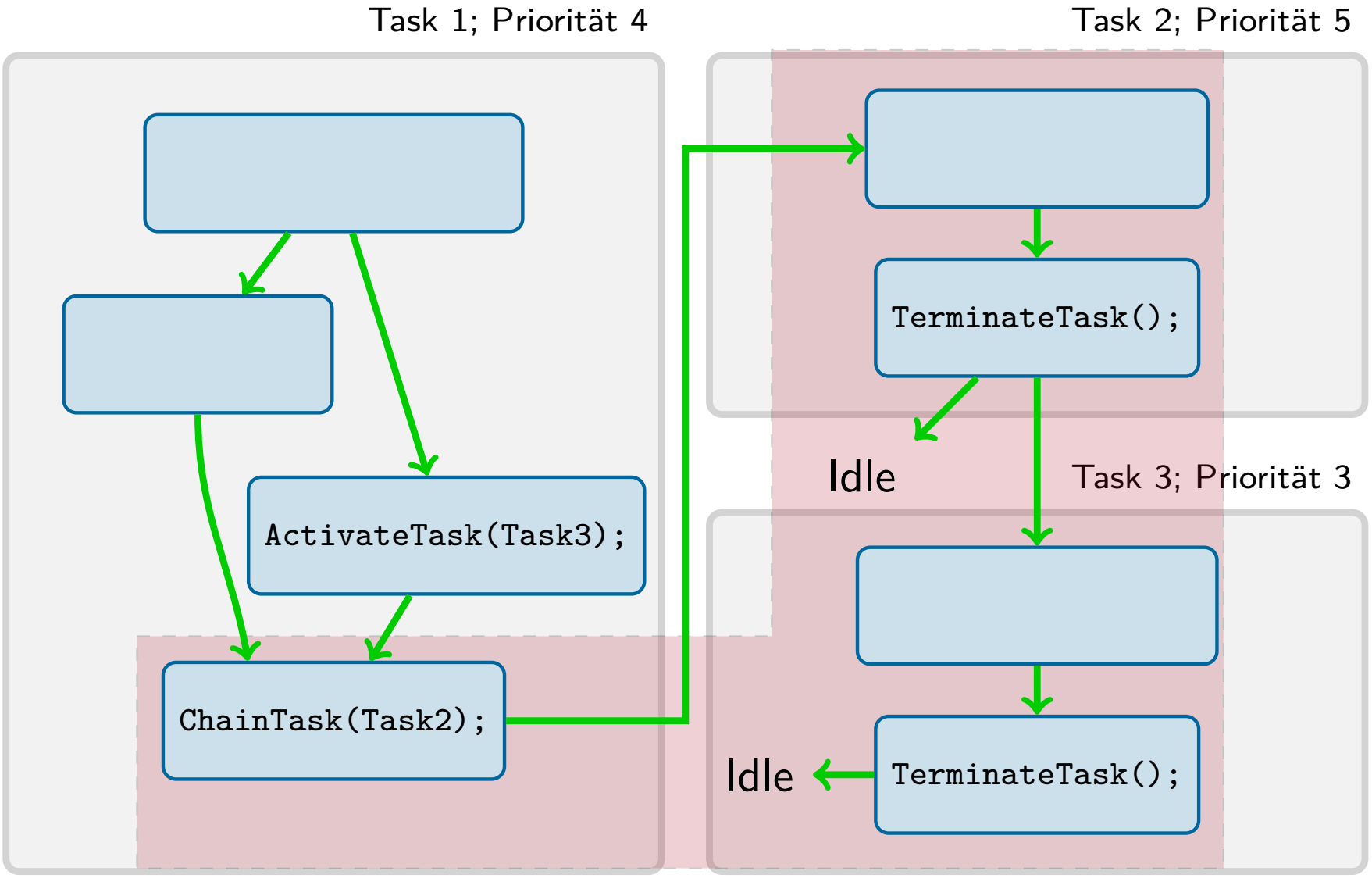
- Feingranulare Analyse von OSEK Systemen
  - Globaler Kontrollflussgraph beschreibt die Anwendung–OS Interaktion
  - Ausführungsreihenfolge für alle Blöcke des Systems
  - Zusätzliche Informationen über den Systemzustand
- Maßschneiderung von Anwendung und Betriebssystem
  - Erzeugung von maßgeschneiderten Kernelfragmenten
    $\Rightarrow$ 53 Prozent niedrigere Kernlaufzeit
  - Überwachung statischer Eigenschaften des Systems
    $\Rightarrow$ 35 Prozent niedrigere SDC-Rate gegenüber bisher bestem *d*OSEK
- Weitere mögliche Anwendungen
  - Verbesserte Analyse der worst-case execution time
  - Informationen für den Übersetzer
  - Ersetzung des OS durch eine Zustandsmaschiene

Beistandsfolien

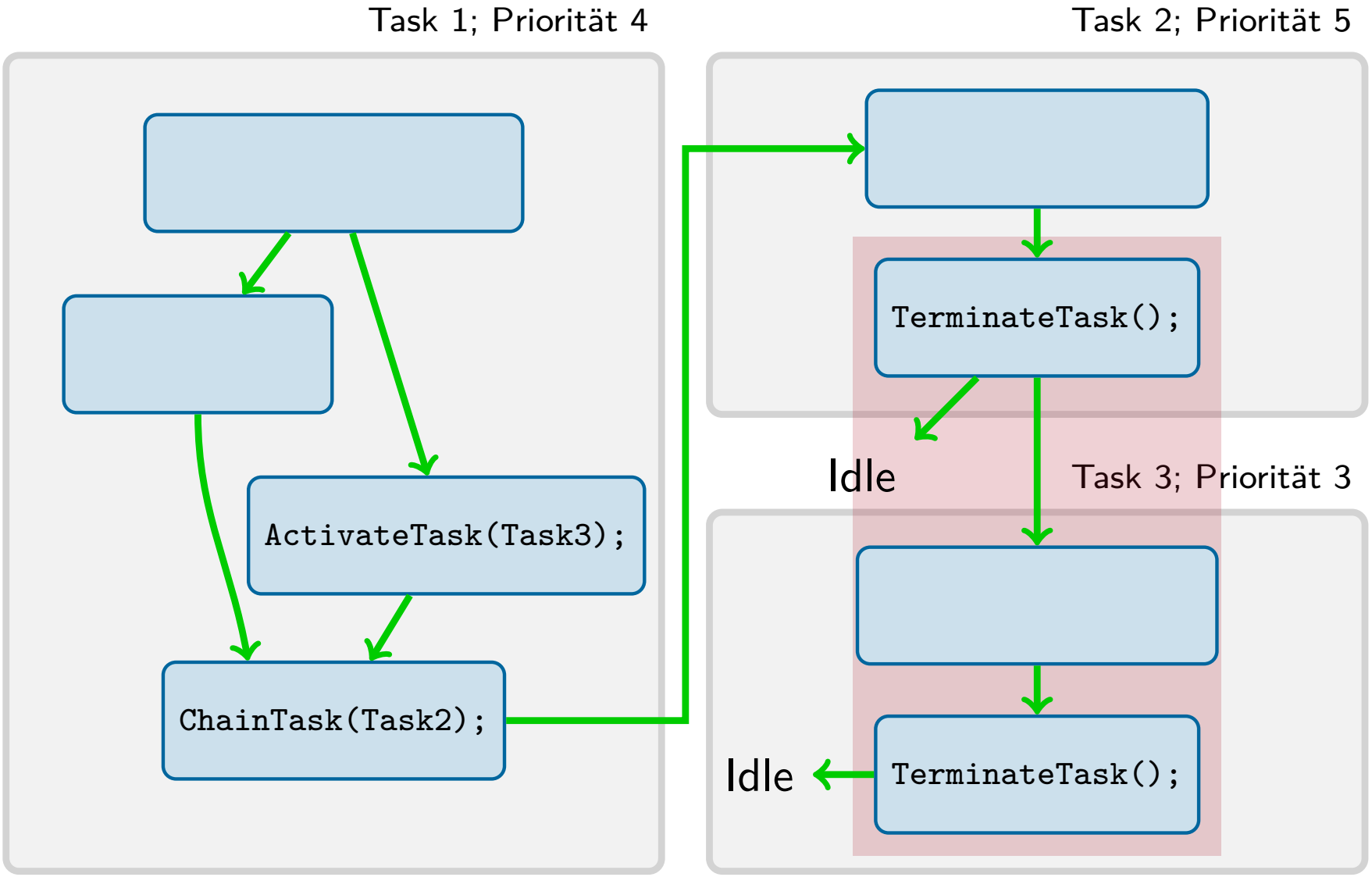
Maßschneiderung des Systems

- Spezialisierung von Systemaufrufen *Laufzeit*
- Zusicherungen über den Zustand des Systems *Fehlertoleranz*
- Kontrollflussüberwachung durch Dominatorregionen *Fehlertoleranz*
  - Regionen die durch einen Knoten betreten werden müssen.
  - Im Eingangsknoten wird eine Marke gesetzt.
  - Diese Marke muss in jedem Knoten der Region vorhanden sein.

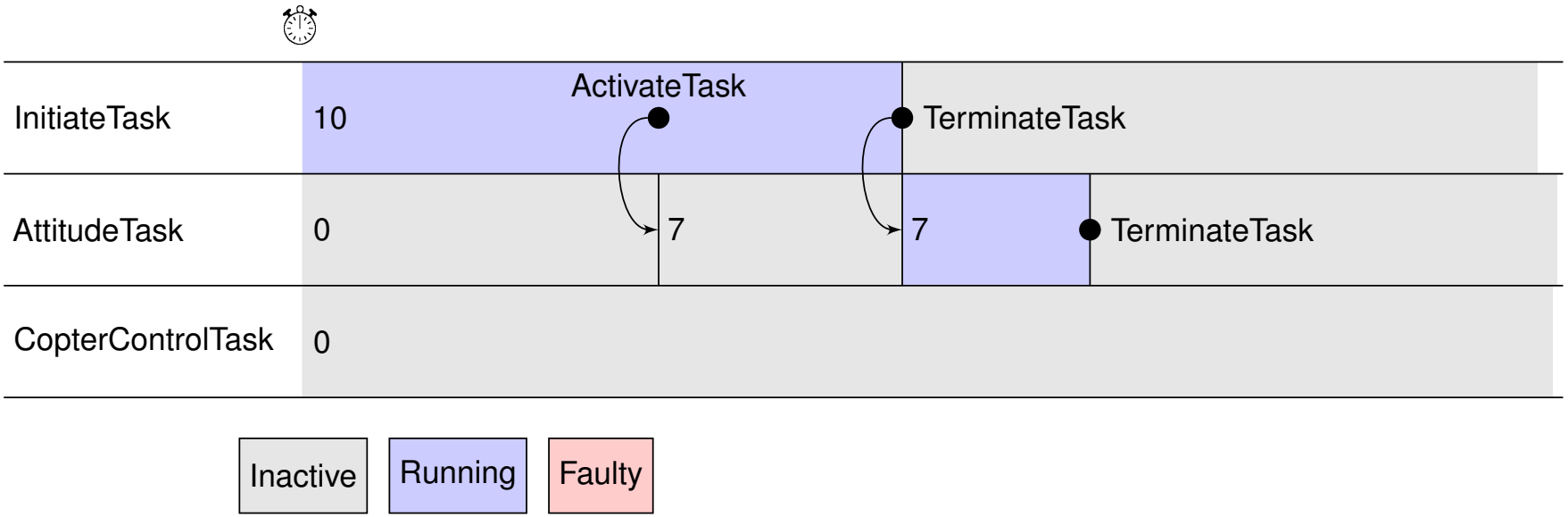
Control Flow Monitoring with Dominator Regions



Control Flow Monitoring with Dominator Regions

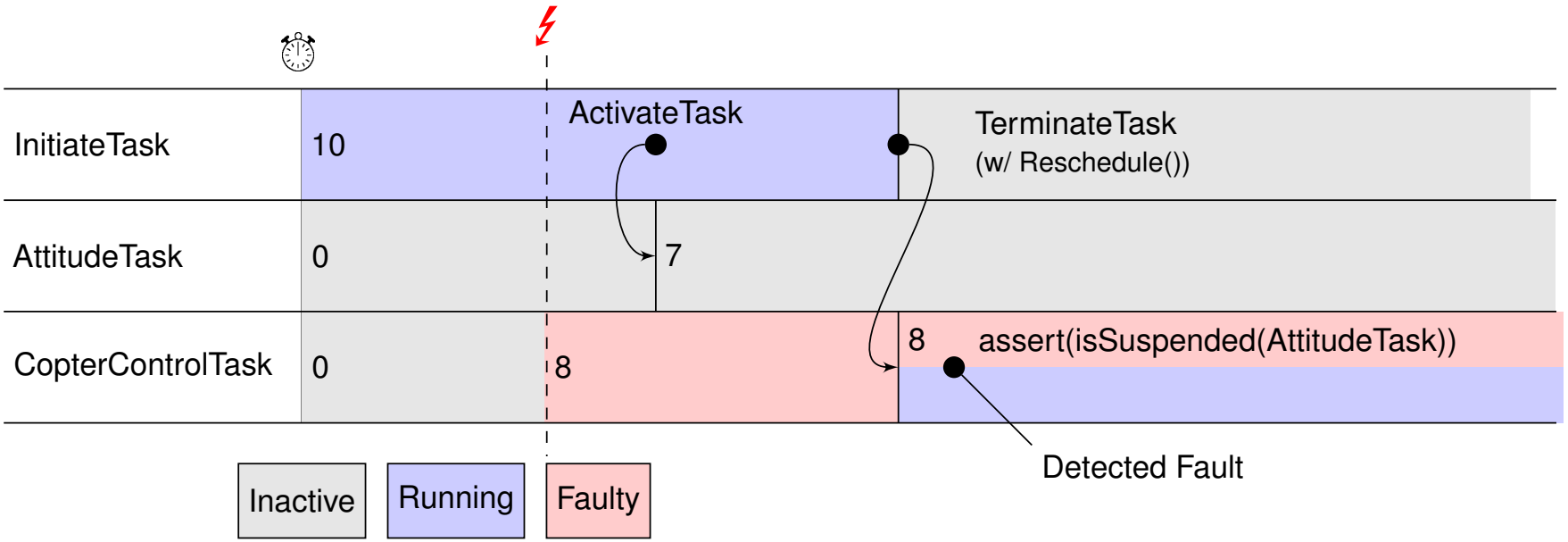


Anomaly in SDC Rate – Golden Run



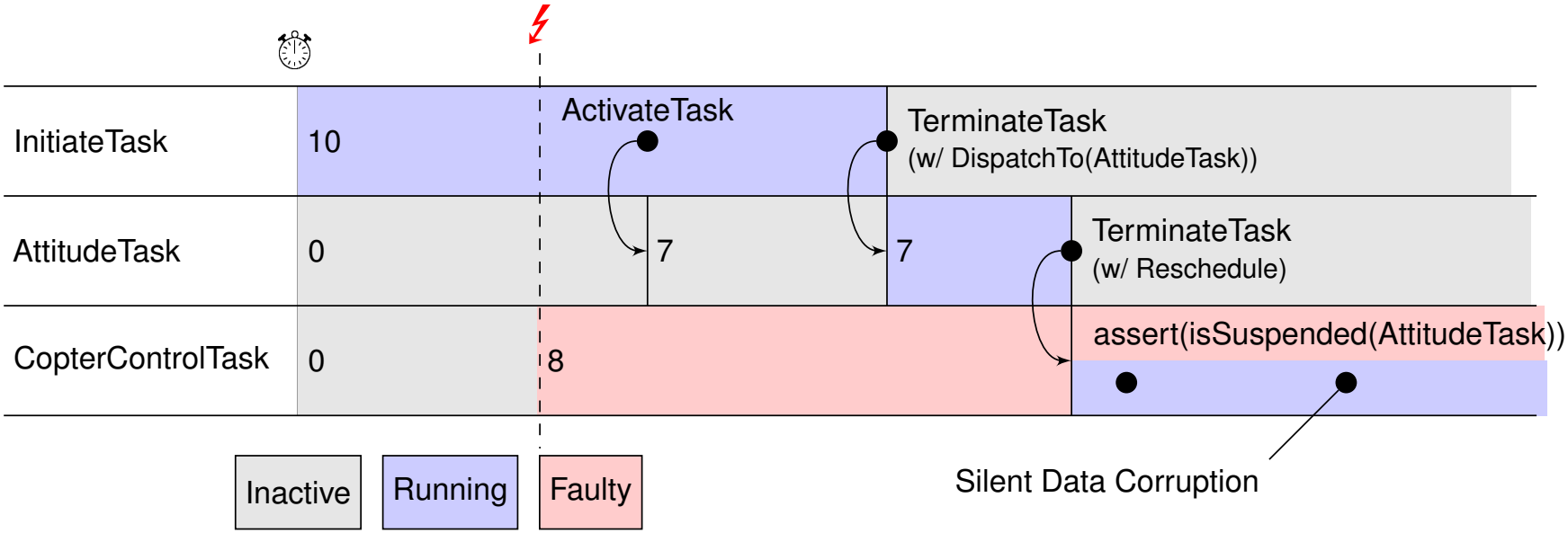
- Scenario with 3 Tasks (Specialization + Assertions)
  - CopterControlTask has the lowest Priorität
  - InititeTask activates the lower-priority AttitudeTask

Anomaly in SDC Rate – with Asserts



- A fault occurs in the scheduler; A task is activated.
  - CopterControlTask is scheduled, since it has priority 8.
  - Because it is the lowest-priority task, it contains system-state asssertions.
  - The fault is detected ⇒ No SDC!

Anomaly in SDC Rate – with Assert + Specialization



- System Call Specialization removes flexibility.
- The fault effect is delayed
  - The assertion cannot catch the fault anymore, because AttitudeTask was scheduled although it was not the highest-priority task.
  - A SDC occurs.