Betriebssysteme (BS)

VL 8.2 – Koroutinen und Fäden – Fortsetzungsmodell

Volkmar Sieh / **Daniel Lohmann**

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 20 – 14. Dezember 2020

https://www4.cs.fau.de/Lehre/WS20/V_BS

Agenda

Grundbegriffe: Routine, Kontrollfluss

- Routine: eine endliche Sequenz von Anweisungen
  - z. B. die Funktion f
  - Sprachmittel fast aller Programmiersprachen
  - wird ausgeführt durch (Routinen-)Kontrollfluss
- (Routinen-)Kontrollfluss: eine Routine in Ausführung
  - Ausführung und Kontrollfluss sind synonme Begriffe
  - z. B. die Ausführung $<f>$ der Funktion f
    - beginnt bei Aktivierung mit der ersten Anweisung von f

Zwischen Routinen und Ausführungen besteht eine Schema–Instanz Relation. Zur klaren Unterscheidung werden die Instanzen ($\mapsto$ Ausführungen) deshalb hier in spitzen Klammern gesetzt:

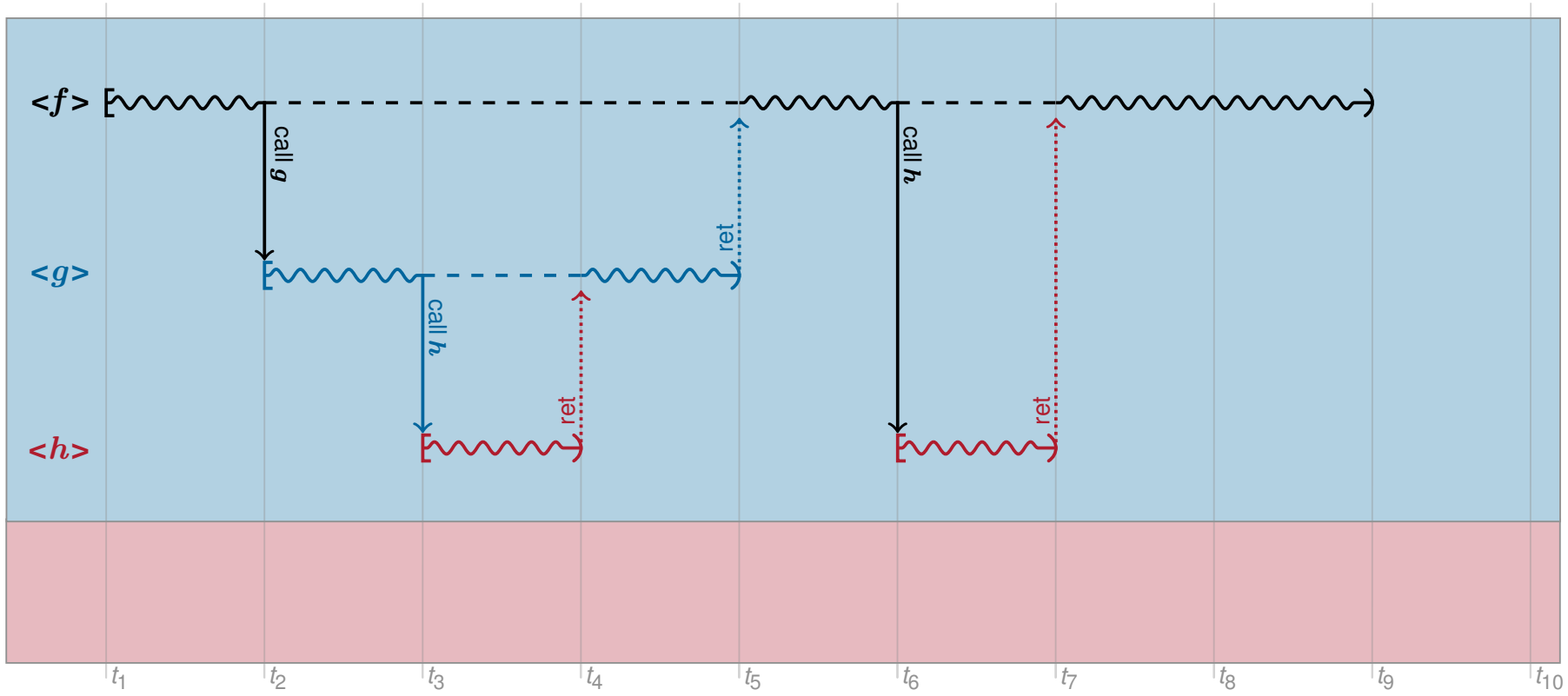
$<f>$, $<f'>$, $<f''>$ sind Ausführungen von f.

Grundbegriffe: Routine, Kontrollfluss

- Routinen-Kontrollflüsse werden erzeugt, gesteuert, und zerstört mit speziellen Elementaroperationen
  - <f> *call* g (Ausführung <f> erreicht Anweisung `call g`)
    1. **erzeugt** neue Ausführung <g> von g
    2. **suspendiert** die Ausführung <f>
    3. **aktiviert** die Ausführung <g>
       ($\rightsquigarrow$ erste Anweisung wird ausgeführt)
  - <g> *ret* (Ausführung <g> erreicht Anweisung `ret`)
    1. **suspendiert** die Ausführung <g>
    2. **zerstört** die Ausführung <g>
    3. **reaktiviert** die Ausführung des Vater-Kontrollflusses (z. B. <f>)

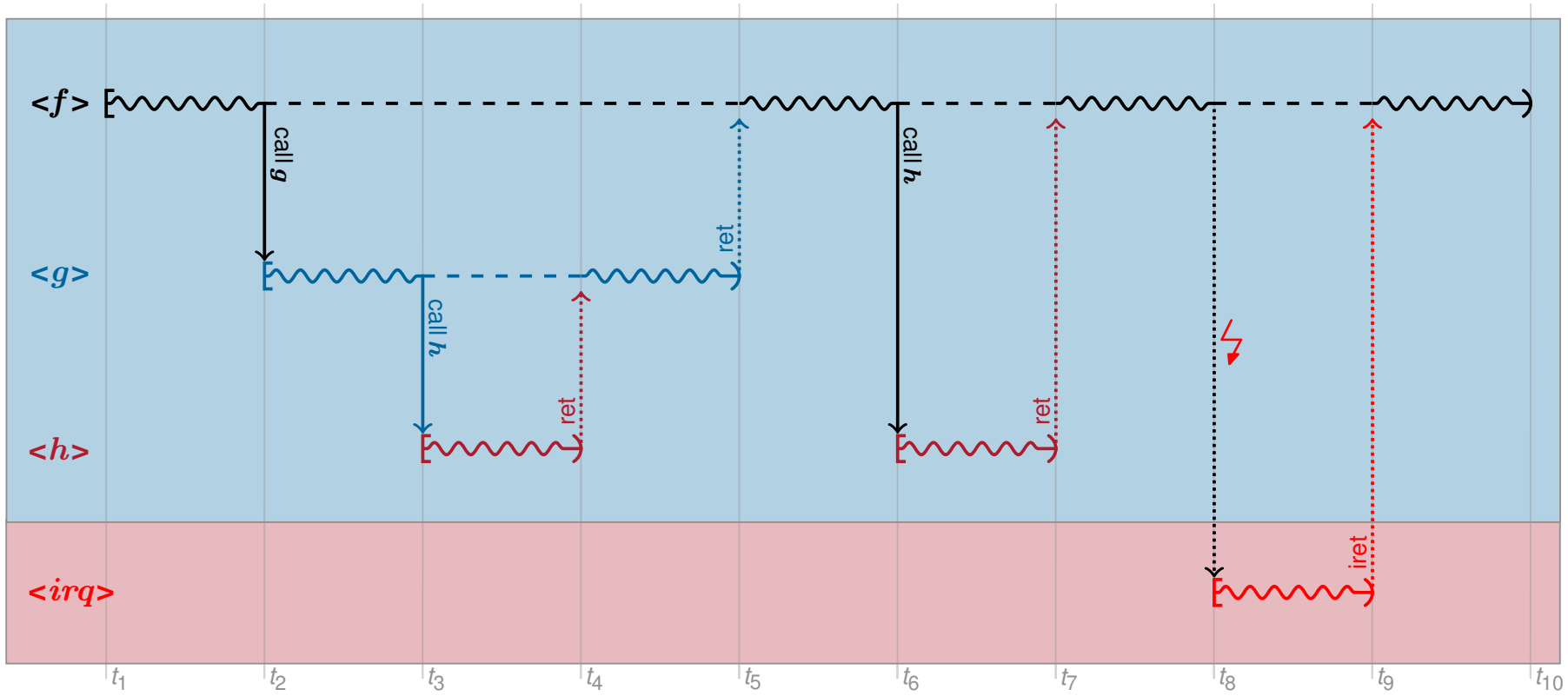
Routinen $\mapsto$ asymmetrisches Fortsetzungsmodell

- Routinen-Kontrollflüsse bilden eine Fortsetzungshierarchie
  - Vater–Kind Relation zwischen Erzeuger und Erzeugtem
- Aktivierte Kontrollflüsse werden nach LIFO fortgesetzt
  - Der zuletzt aktivierte Kontrollfluss terminiert immer zuerst
  - Vater wird erst fortgesetzt, wenn Kind terminiert



Routinen ↦ asymmetrisches Fortsetzungsmodell

- Das gilt auch bei Unterbrechungen
  - $<f>$ ↯ irq ist wie *call*, nur implizit
- Unterbrechungen können als implizit erzeugte und aktivierte Routinen-Ausführungen verstanden werden



Grundbegriffe: Koroutine

- Koroutine (engl. *Coroutine*): verallgemeinerte Routine [1]
  - erlaubt zusätzlich: expliziten Austritt und Wiedereintritt
  - Sprachmittel *einiger* Programmiersprachen
    - z. B. Modula-2, Simula-67, Stackless Python
  - wird ausgeführt durch Koroutinen-Kontrollfluss
- Koroutinen-Kontrollfluss: eine Koroutine in Ausführung
  - Kontrollfluss mit eigenem, unabhängigen Zustand
    - mindestens Programmzähler (PC)
    - zusätzlich je nach (zu unterstützendem) Compiler / ABI / ISA: weitere Register, Stapel, ...
    - Im Prinzip ein eigenständiger Faden (engl. *Thread*) – **dazu später mehr**

Koroutinen und Koroutinen-Kontrollflüsse stehen ebenfalls in einer Schema–Instanz Relation.

In der Literatur ist diese Unterscheidung unüblich $\rightsquigarrow$ Koroutinen-Kontrollflüsse werden (vereinfacht) ebenfalls als Koroutinen bezeichnet.

Grundbegriffe: Koroutine

- Koroutinen-Kontrollflüsse werden erzeugt, gesteuert, und zerstört über zusätzliche Elementaroperationen
  - *create* g
    1. **erzeugt** neue Korotinen-Ausführung <g> von g
  - <f> *resume* <g>
    1. **supendiert** die Koroutinen-Ausführung <f>
    2. **(re-)aktiviert** die Koroutinen-Ausführung <g>
  - *destroy* <g>
    1. **zerstört** die Koroutinen-Ausführung <g>

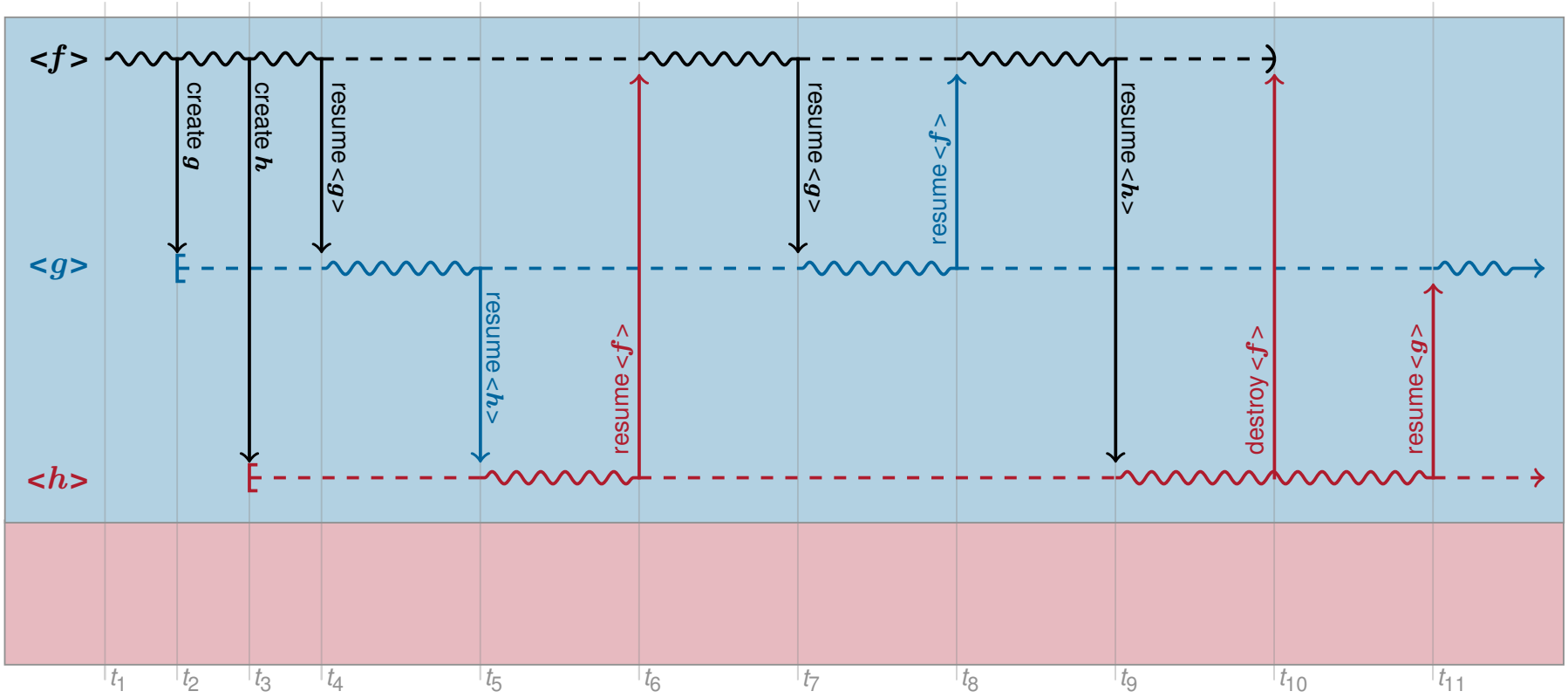
Unterschied zu Routinen-Kontrollflüssen: [SP, C 10-8]

Aktivierung und Reaktivierung sind
zeitlich entkoppelt von Erzeugung und Zerstörung.

$\rightsquigarrow$ Koroutinen sind **echt mächtiger** als Routinen.

Koroutinen ↦ symmetrisches Fortsetzungsmodell

- Koroutinen-Kontrollflüsse bilden eine Fortsetzungsfolge
  - Koroutinenzustand bleibt über Ein-/Austritte hingweg erhalten
- Alle Koroutinen-Kontrollflüsse sind gleichberechtigt
  - kooperatives Multitasking
  - Fortsetzungsreihenfolge ist beliebig



Koroutinen und Programmfäden

- Koroutinen-Kontrollflüsse werden oft auch bezeichnet als
  - kooperative Fäden (engl. *cooperative Threads*)
  - Fasern (engl. Fibers)
- Das ist im Prinzip richtig, die Begriffe entstammen jedoch aus verschiedenen Welten
  - Koroutinen-Unterstützung ist historisch (eher) ein **Sprach**merkmal
  - Mehrfädigkeit ist historisch (eher) ein **Betriebssystem**merkmal
  - Die Grenzen sind fließend
    - *Sprachfunktion — (Laufzeit-)Bibliothekfunktion — Betriebssystemfunktion*
- Wir verstehen Koroutinen als technisches Konzept
  - um Mehrfädigkeit im BS zu implementieren
  - insbesondere später auch nicht-kooperative Fäden