Verklemmungen

Erinnerung: Betriebsmittel

Wiederverwendbare Betriebsmittel

- Werden von Prozessen für eine bestimmte Zeit belegt und anschließend wieder freigeben
- Beispiele: CPU, Speicher, E/A-Geräte
- Schema: Gegenseitiger Ausschluss

Konsumierbare Betriebsmittel

- Werden im laufenden System erzeugt/konsumiert
- Beispiele: Signale, Nachrichten, IO-Ströme
- Schema: Einseitige Synchronisation

Zugbeispiel

Zwei Züge (Zug A und Zug B) bewegen sich auf einer eingleisigen Strecke mit zwei Blockabschnitten. Jeder Zug benötigt beide Blockabschnitte, um die Strecke sicher passieren zu können. Die Blockabschnitte werden durch Signale gesichert und können jeweils nur von einem Zug gleichzeitig belegt werden.

Ein Zug kann nur dann weiterfahren, wenn er beide Blockabschnitte gleichzeitig belegen kann, da er sonst nicht sicher die Strecke passieren kann. Nachdem ein Zug einen Abschnitt durchfahren hat, gibt er die Blockabschnitte wieder frei, damit der andere Zug passieren kann. Da die Züge pünktlich bleiben wollen, warten sie auf die Freigabe der Blockabschnitte, falls diese gerade belegt sind.

Die Zugführer in diesem Szenario haben keine Kenntnis von Verklemmungen. Daher gibt es keine Regelung, in welcher Reihenfolge die Blockabschnitte angefordert werden müssen. Der eine Zug fordert immer zuerst Blockabschnitt 1 und danach Blockabschnitt 2 an, der andere Zug macht es genau umgekehrt.

Vorbedingungen für Verklemmungen

- Was sind die Betriebsmittel im Beispiel?
- Blockabschnitte

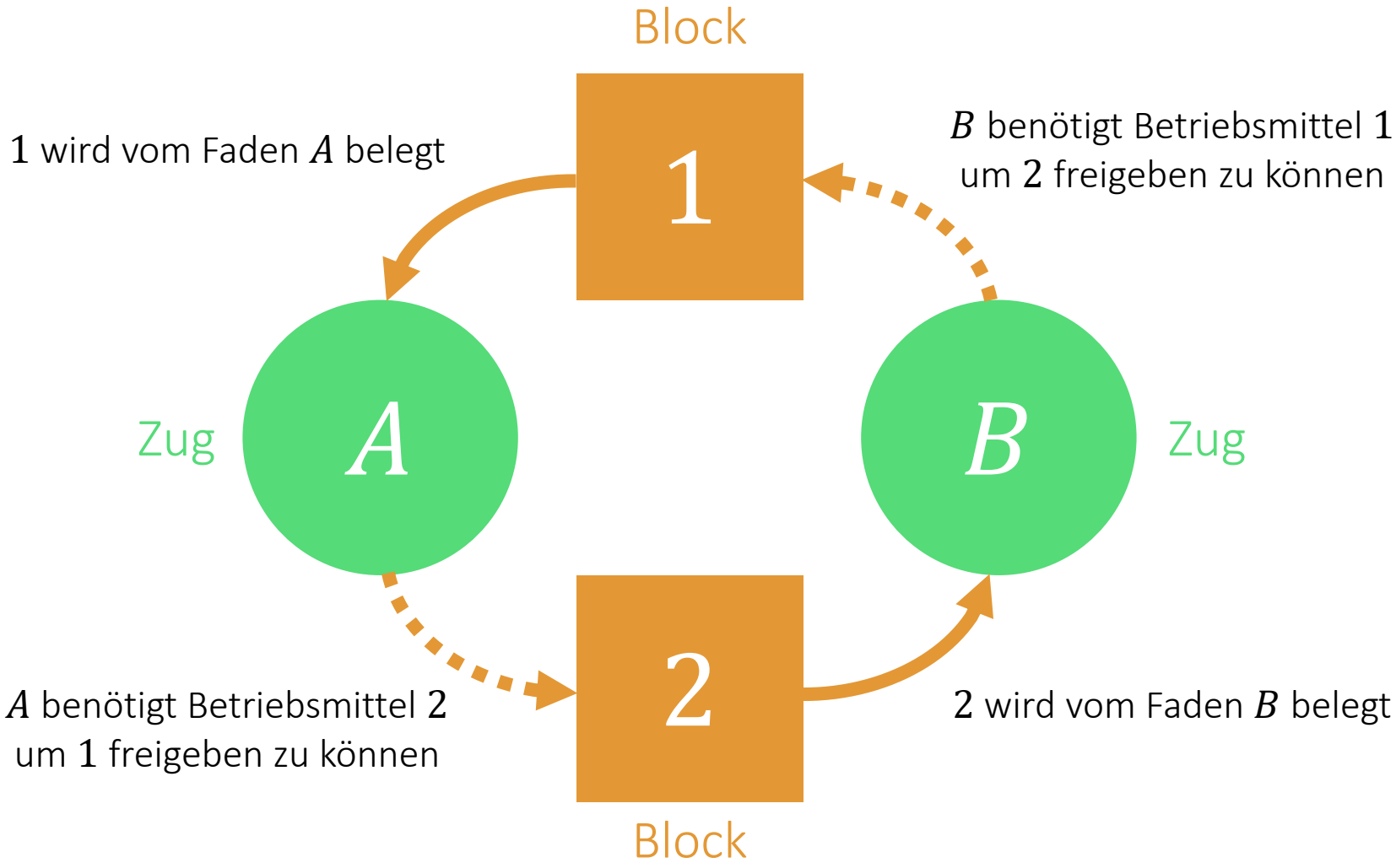
Vorbedingungen für Verklemmungen

- Damit Verklemmungen entstehen können, müssen bestimmte Bedingungen erfüllt sein:
- **Gegenseitiger Ausschluss**: Nur ein Zug darf sich in einem Block befinden
- **Halten und Warten**: Ein Zug muss den nächsten Block belegen, bevor er den aktuellen verlassen kann
- **Keine Rückforderbarkeit**: Ein Zug kann nicht ohne weiteres vom Gleis verdrängt werden

Vorbedingungen für Verklemmungen

- Damit wirklich eine Verklemmung entsteht, muss zur Laufzeit noch eine weitere Bedingung eintreten:
- **Zirkuläres Warten**: Aktivitätsträger warten wechselseitig auf Betriebsmittel

Zugriffsgraph (Zirkuläres warten)



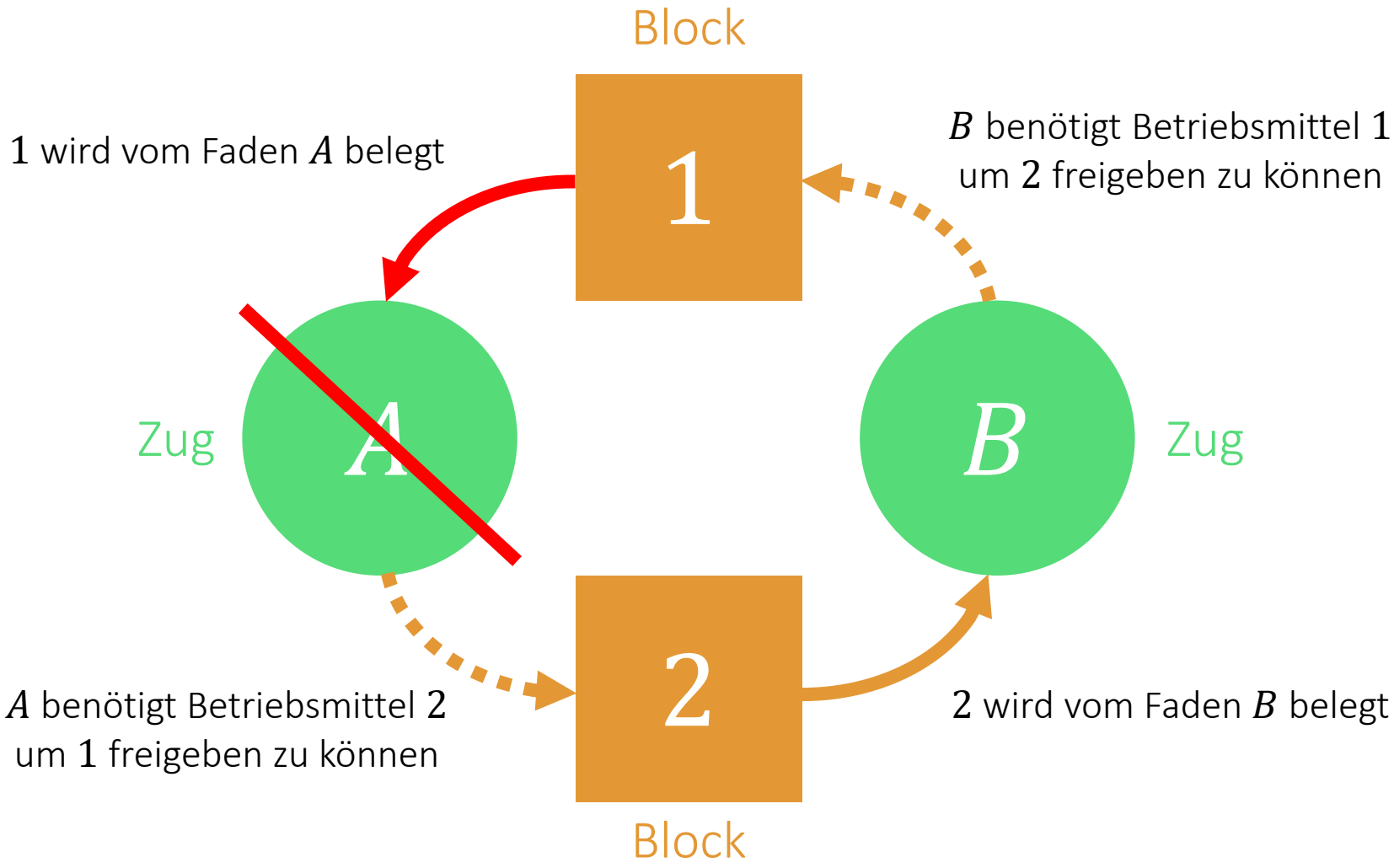
Verklemmung erkennen

```
Train B: Now requesting block2.
Train A: Acquired block1.
Train A: Now requesting block2.
Train B: Acquired block2.
Train B: Now requesting block1.
```

- Nach dieser Ausgabe hält A den block1 und B den block2
- Beide warten jeweils auf den anderen Block
- Zur Erkennung muss geprüft werden,
  ob genau diese Belegung gilt

```
current_status_a == request_block2 && current_status_b == request_block1
```

Verklemmung erkennen



Verklemmung auflösen

- Wie lassen sich allgemein Verklemmungen auflösen?
- Einen Aktivitätsträger abbrechen und neu aufsetzen, oder anderweitig in einen sicheren Zustand zurücksetzen
- Wie lassen sich Im Beispiel Verklemmungen auflösen?
- Ein Zug könnte Zurücksetzen und auf ein Abstellgleis ausweichen, bis der andere Zug abgefahren ist

Vorbeugung

- Wie lassen sich Verklemmungen vorbeugen?
- Alle Züge fahren in dieselbe Richtung
- Belegungsreihenfolge ist immer gleich
- Vorbeugung ≠ Vermeidung ≠ Auflösung

Verklemmung erkennen

```
void check_for_deadlocks(void) {

   printf("Dispatcher: Checking the trains.\n");

   if (sem_wait(&status) == -1) {/* … */}
   enum STATUS current_status_a = train_a_status;
   enum STATUS current_status_b = train_b_status;
   if (sem_post(&status) == -1) {/* … */}

   if (current_status_a == request_block2 && current_status_b == request_block1) {
      printf("Dispatcher: Deadlock detected!\n");
   } else {
      printf("Dispatcher: Everything OK.\n");
   }
}
```

Verklemmungsauflösung (im Beispiel)

```
void check_for_deadlocks(void) {

   printf("Dispatcher: Checking the trains.\n");

   if (sem_wait(&status) == -1) {/* … */}
   enum STATUS current_status_a = train_a_status;
   enum STATUS current_status_b = train_b_status;
   if (sem_post(&status) == -1) {/* … */}

   if (current_status_a == request_block2 && current_status_b == request_block1) {
      if (pthread_cancel(train_a_thread) == -1) {/* … */}
      if (pthread_join(train_a_thread, NULL) == -1) {/* … */}
      if (sem_post(&block1) == -1) {/* … */}
      if (pthread_create(&train_a_thread, NULL, &train_a, NULL) != 0) {/* … */}
   } else {
      printf("Dispatcher: Everything OK.\n");
   }
}
```