Betriebssysteme

Tutorium 7

Philipp Kirchhofer
philipp.kirchhofer@student.kit.edu
http://www.stud.uni-karlsruhe.de/~uxbtt/

Lehrstuhl Systemarchitektur
Universität Karlsruhe (TH)

9. Dezember 2009

Was machen wir heute?

1. Tutorien Übungsblatt
   - RAGs and WFGs
   - Prerequisites for Deadlocks
   - Deadlock Avoidance
   - Searching for Deadlocks

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.a

Welche Knotentypen kommen in einem Resource Allocation Graph (RAG) vor?

Antwort

- Prozesse
- Ressourcen

Prozesse werden in der grafischen Darstellung rund und Ressourcen rechteckig dargestellt.

Frage 7.1.b

Wie werden Ressourcen mit mehreren Instanzen dargestellt?

Antwort

Für jede Instanz einer Ressource wird ein Punkt in den Knoten gezeichnet.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.c

Was ist eine Anfragekante (request edge)?

Antwort

Gerichtete Kante, die von einem Prozessknoten auf einen Ressourcenknoten zeigt.

Bedeutung: Der Prozess braucht eine Instanz der Ressource für die Bearbeitung seiner Aufgabe.

Frage 7.1.d

Was ist eine Zuweisungskante (assignment edge)?

Antwort

Gerichtete Kante, die von einer Instanz einer Ressource auf einen Prozessknoten zeigt.

Bedeutung: Eine Instanz der Ressource wurde dem Prozess zugewiesen.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.e

Ergibt sich aus einem Zyklus im RAG zwangsläufig ein Deadlock?

Antwort

Nein, da ein Zyklus im RAG nur dann zu einem Deadlock führt, wenn jede beteiligte Ressource nur eine Instanz hat. Bei Ressourcen mit mehreren Instanzen muss es nicht zwangsläufig zu einem Deadlock kommen.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.f

Was ist ein Wait For Graph (WFG)?

Antwort

Ein WFG ist eine Variante des RAG, in der alle Ressourcenknoten entfernt wurden. In einem WFG zeigt eine gerichtete Kante von Prozessknoten P_i zu Prozessknoten P_j an, dass Prozess P_i auf eine Ressource wartet, die Prozess P_j zur Zeit hält.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.g

Wie kann ein WFG aus einem RAG erstellt werden?

Antwort

1. Kante von P_i nach P_j eintragen, wenn im RAG eine Kante von P_i zu einer beliebigen Ressource R und von R zu P_j existiert
2. Alle Ressourcenknoten aus dem RAG entfernen

Frage 7.1.h

Wie kann ein RAG aus einem WFG erstellt werden?

Antwort

Aus einem WFG kann kein eindeutiger RAG erstellt werden, da die Verbindung von Prozessen zu Ressourcen nicht rekonstruiert werden kann.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.i

Kann ein WFG aus einem RAG mit mehreren Instanzen pro Ressource erstellt werden?

Antwort

Nein, da keine Eindeutigkeit bei der Erstellung des WFG besteht.

Tutorien Übungsblatt - RAGs and WFGs

Frage 7.1.j

Welche Einsatzmöglichkeit besteht für einen WFG?

Antwort

Deadlockerkennung (Deadlock Detection)

Das Entfernen der Ressourcenknoten senkt den Speicher- und Suchaufwand bei der späteren Deadlockerkennung.

Tutorien Übungsblatt - Prerequisites for Deadlocks

Wiederholung: Bedingungen für Deadlock

Welche Bedingungen müssen erfüllt sein, damit ein Deadlock auftreten kann?

Antwort

- No Preemption
  Die Betriebsmittel werden ausschließlich durch die Prozesse freigegeben
- Hold and Wait
  Die Prozesse fordern Betriebsmittel an, behalten aber zugleich den Zugriff auf andere
- Mutual Exclusion
  Der Zugriff auf die Betriebsmittel ist exklusiv
- Circular Wait
  Mindestens zwei Prozesse besitzen bezüglich der Betriebsmittel eine zirkuläre Abhängigkeit

Tutorien Übungsblatt - Prerequisites for Deadlocks

Frage 7.2

Wie können Deadlocks vermieden werden?

Antwort

- No Preemption
  Erlauben, dass Betriebsmittel an einen anderen Prozess vergeben werden können (Speichern und Laden des Zustands bei Zuweisungswechsel)
- Hold and Wait
  Prozesse fordern alle benötigten Betriebsmittel atomar am Programmanfang an, danach keine weitere Zuweisung
- Mutual Exclusion
  Gemeinsam genutztes Objekt erhält exklusiven Zugriff auf Betriebsmittel (Multiplexing, Spooler bei Drucker)
- Circular Wait
  Ordnen der benötigten Ressourcen in einer beliebigen Reihenfolge und anschließende Vergabe in aufsteigender Reihenfolge

Tutorien Übungsblatt - Deadlock Avoidance

System

Gegeben sei folgendes System mit 4 Ressourcen und 5 Prozessen:

Allocation:

	R_1	R_2	R_3	R_4
P_1	0	0	1	2
P_2	1	0	0	0
P_3	1	3	5	4
P_4	0	6	3	2
P_5	0	0	1	4

Max:

	R_1	R_2	R_3	R_4
P_1	0	0	1	2
P_2	1	7	5	0
P_3	2	3	5	6
P_4	0	6	5	2
P_5	0	6	5	6

Available:

R_1	R_2	R_3	R_4
1	5	2	0

Tutorien Übungsblatt - Deadlock Avoidance

Frage 7.3.a

Wieviele Ressourcen benötigen die Prozesse noch?

Antwort

Need = Max - Allocation =

	R_1	R_2	R_3	R_4
P_1	0	0	0	0
P_2	0	7	5	0
P_3	1	0	0	2
P_4	0	0	2	0
P_5	0	6	4	2

Tutorien Übungsblatt - Deadlock Avoidance

Frage 7.3.b

Was ist ein sicherer Zustand?

Antwort

Ein sicherer Zustand ist ein Zustand des Systems, in dem jeder Prozess zu Ende laufen kann, selbst wenn jeder der Prozesse den maximalen Ressourcenbedarf hat. Noch nicht laufende Prozesse, deren Ressourcenbedarf nicht sofort gedeckt werden kann, müssen warten, bis laufende Prozesse ihre Ressourcen freigegeben haben.

Tutorien Übungsblatt - Deadlock Avoidance

Frage 7.3.c

Ist das System in einem sicheren Zustand?

Antwort

Ja. Mit den verfügbaren Ressourcen $(1, 5, 2, 0)$ kann entweder Prozess P_1 oder P_4 laufen. Nachdem P_4 gelaufen ist kann mit den dann verfügbaren Ressourcen $(1, 11, 5, 2)$ jeder andere Prozess erfolgreich ausgeführt werden.

Tutorien Übungsblatt - Deadlock Avoidance

Frage 7.3.d

Von Prozess P_2 trifft die Ressourcenanfrage $(0, 4, 2, 0)$ ein. Kann diese Anfrage sofort umgesetzt werden?

Antwort

Es muss geprüft werden, ob der Endzustand ein sicherer Zustand ist. Dazu nimmt man an, dass die Anfrage umgesetzt wurde und die Ressourcen Prozess P_2 zur Verfügung stehen.

$\text{Available} = \text{Available} - \text{Request} = (1, 5, 2, 0) - (0, 4, 2, 0) = (1, 1, 0, 0)$
$\text{Allocation}_2 = \text{Allocation}_2 + \text{Request} = (1, 0, 0, 0) + (0, 4, 2, 0) = (1, 4, 2, 0)$
$\text{Need}_2 = \text{Need}_2 - \text{Request} = (0, 7, 5, 0) - (0, 4, 2, 0) = (0, 3, 3, 0)$

Tutorien Übungsblatt - Deadlock Avoidance

Antwort

Dadurch ergibt sich folgender Systemzustand:

Allocation:

	R_1	R_2	R_3	R_4
P_1	0	0	1	2
P_2	1	4	2	0
P_3	1	3	5	4
P_4	0	6	3	2
P_5	0	0	1	4

Need:

	R_1	R_2	R_3	R_4
P_1	0	0	0	0
P_2	0	3	3	0
P_3	1	0	0	2
P_4	0	0	2	0
P_5	0	6	4	2

Mit den verfügbaren Ressourcen $(1, 1, 0, 0)$ kann Prozess P_1 abgeschlossen werden, danach stehen die Ressourcen $(1, 1, 1, 2)$ zur Verfügung. Anschließend kann Prozess P_3 abgeschlossen werden, es stehen dann die Ressourcen $(2, 4, 6, 6)$ zur Verfügung. Mit diesen verfügbaren Ressourcen können die Prozesse P_2 und P_4 abgeschlossen werden, mit den danach verfügbaren Ressourcen $(3, 14, 11, 8)$ kann der letzte Prozess P_5 abgeschlossen werden.

Das System befindet sich also in einem sicheren Zustand.

Tutorien Übungsblatt - Searching for Deadlocks

Programm

```
Spinlock s1, s2, s3 = FREE;
int counter = 0;


Thread1() {
    if ( counter == 0 ) {
        lock(s1);
        counter++;
        unlock(s1);
    }
    lock(s2);
    lock(s3);
    // update some more data
    unlock(s3);
    unlock(s2);
}
```

```
Thread2() {
    lock(s3);
    counter++;
    // update some data
    if ( counter == 2 ) {
        lock(s2);
        // update some more data
        unlock(s2);
    }
    lock(s1);
    // update even more data
    unlock(s3);
    unlock(s1);
}
```

Tutorien Übungsblatt - Searching for Deadlocks

Frage 7.4.a

Können Race Conditions auftreten?

Antwort

Ja. Die Variable „counter“ wird von beiden Threads nicht atomar verändert und ist durch zwei verschiedene Spinlocks geschützt.

Frage 7.4.b

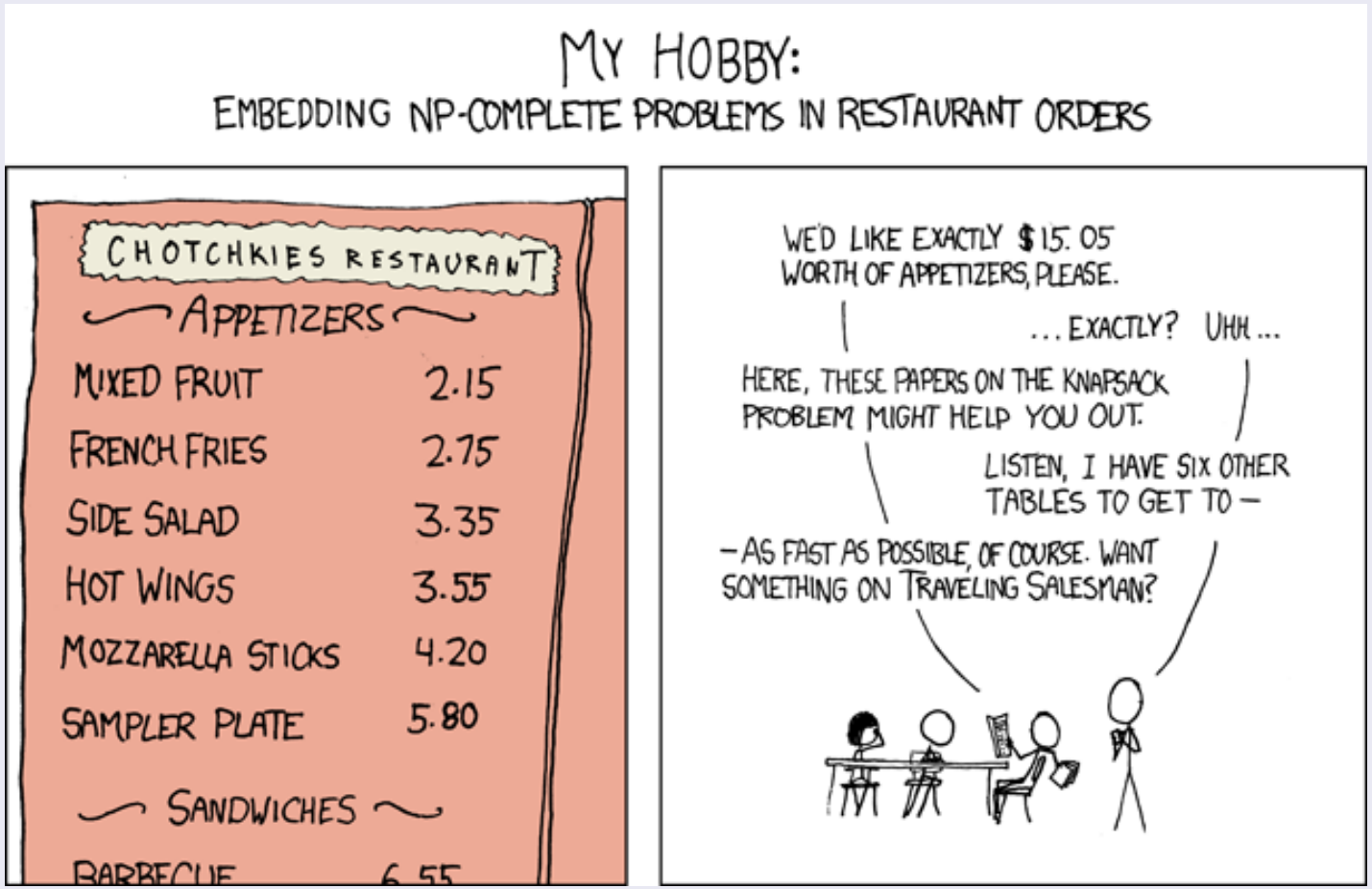
Kann ein Deadlock auftreten?

Antwort

Ja. Durch Preemption kann ein Deadlock entstehen.

Fragen & Kommentare?

Ende



xkcd: NP-Complete