Freie Universität Berlin

Prof. Dr. Margarita Esponda

Nichtsequentielle Programmierung, SoeSe 2017

Übungsblatt 3

TutorIn: Lilli Walter
Tutorium 6

Boyan Hristov, Sergelen Gongor

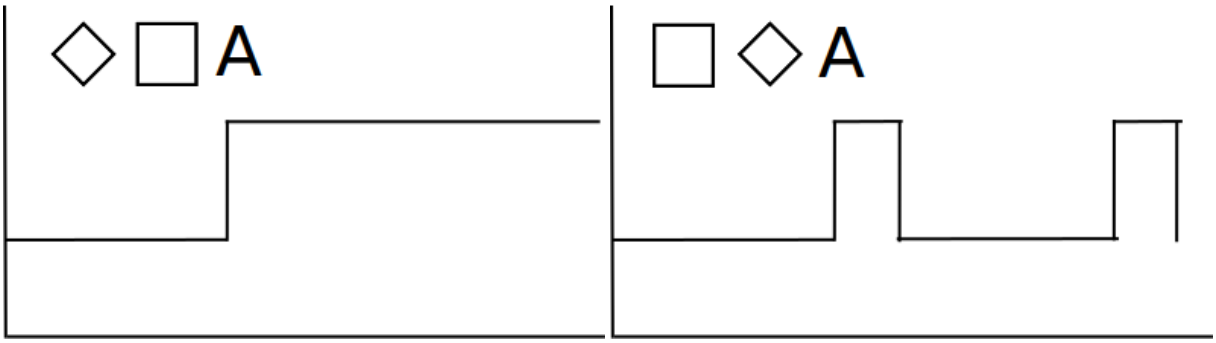
23. Mai 2017

Link zum Git Repository: https://github.com/BoyanH/FU-Berlin-ALP4/tree/master/Solutions/Homework3

1 Aufgabe

$$\Diamond\Box A \Leftrightarrow \Box\Diamond A \tag{1}$$
$$\Leftrightarrow \exists j \geq i : \Box A \text{ in } S_j \Leftrightarrow \forall j' \geq i : \Diamond A \text{ in } S_{j'} \tag{2}$$
$$\Rightarrow \forall j' \geq i : \Diamond A \text{ in } S_{j'} \Rightarrow \exists j \geq i : \Box A \text{ in } S_j \tag{3}$$
$$\Leftrightarrow \exists j \geq i : (\Diamond A \Rightarrow \Box A) \text{ in } S_j \tag{4}$$

3 $\Leftrightarrow$ 4: Aber da $\Diamond A$ in alle $j' \geq i$ gilt, dann auch in j! Widerspruch!



2 Aufgabe

$$\Box A \Rightarrow \Diamond B \wedge \Diamond \Box A \tag{5}$$
$$\Leftrightarrow \forall j \geq i A \text{ in } S_j \Rightarrow \exists k \geq i : B \text{ in } S_k \wedge \exists l \geq i \forall p \geq l A \text{ in } S_p \tag{6}$$
$$\Rightarrow \exists l \geq i \exists k \geq l : B \text{ in } S_k \tag{7}$$
$$\Leftrightarrow \exists k \geq l : B \text{ in } S_k \tag{8}$$
$$\Leftrightarrow \Diamond B \tag{9}$$

Informal: Wenn A immer gilt, dann gilt irgendwann B (Implikation) UND in irgendeinem Zustand $S_j, j \geq i$ gilt immer A. Dann wird ab S_j die linke Seite der Implikation gelten, muss also die Rechte auch gelten.

3 Aufgabe

a)

$$(\Box A \wedge \Box B) \Leftrightarrow \Box (A \wedge B) \tag{10}$$
$$\Leftrightarrow \forall k \geq i : A \text{ in } S_k \wedge \forall l \geq i : B \text{ in } S_l \tag{11}$$
$$\Leftrightarrow \forall k \geq i : A \text{ in } S_k \wedge \forall k \geq i : B \text{ in } S_k \tag{12}$$
$$\Leftrightarrow \forall k \geq i : A \wedge B \text{ in } S_k \tag{13}$$
$$\Leftrightarrow (A \wedge B) \tag{14}$$

Informal: Die Distributivität ist hier anhand von triviale Umformungen mit Quantoren bewiesen. Wenn etwas für alle $k \geq i$ und für alle $l \geq i$ gilt, dann ist es egal welche von beiden wir nehmen, da beide von dem selben Wert (i) gelten und dann für ALLE Nachfolger. Wenn dann zwei mit logischem AND verbundene Aussage mit Quantoren gelten, wobei die Quantoren equivalent sind, darf man diese auch in einem Quantor vereinigen.

Noch informaler: Es gilt immer A und es gilt immer B, d.h. zu jedem Zeitpunk gilt A und zu jedem Zeitpunk gilt B. Dann egal wann wir diese beide Aussagen mit einem logischen AND verknüpfen, wird das Resultat immer wahr sein, also es gilt immer A UND B.

b)

$$(\Rightarrow) \tag{15}$$
$$\exists k \geq i : A \text{ in } S_k \vee \exists p \geq i : B \text{ in } S_p \tag{16}$$
$$\text{1. Fall} \tag{17}$$
$$\exists k \geq i : A \text{ in } S_k \Rightarrow \exists k \geq i : A \text{ in } S_k \vee B \text{ in } S_k \tag{18}$$
$$\Leftrightarrow \Diamond (A \vee B) \tag{19}$$
$$\text{2. Fall} \tag{20}$$
$$\exists p \geq i : B \text{ in } S_p \Rightarrow \exists p \geq i : B \text{ in } S_p \vee A \text{ in } S_p \tag{21}$$
$$\Leftrightarrow \Diamond (A \vee B) \tag{22}$$
$$(\Leftarrow) \tag{23}$$
$$\Diamond (A \vee B) \tag{24}$$
$$\Leftrightarrow \exists k \geq i : (A \text{ in } S_k \vee B \text{ in } S_k) \tag{25}$$
$$\Rightarrow \exists k \geq i : A \text{ in } S_k \vee \exists k \geq i : B \text{ in } S_k \tag{26}$$
$$\Leftrightarrow \Diamond A \vee \Diamond B \tag{27}$$

Informal: Dieses Beweis basiert sich darauf, dass $A \equiv (A \vee False)$. Wenn im 1. Fall irgendwan A gilt, dann gilt auch irgendwann $A \vee False$ und deswegen auch $A \vee B$, analog für 2. Fall. In die andere Richtung, wenn irgendwann $A \vee B$ gilt, dann gilt im 1. Fall irgendwann $(A \vee False)$ oder im 2. Fall $(False \vee A)$, wovon man $\Diamond A \vee False$ bzw. $False \vee \Diamond B$ ableiten kann und endlich $\Diamond A \vee \Diamond B$.

4 Aufgabe

1)

$$(p_4 \wedge \Box\neg p_5) \tag{28}$$
$$\Leftrightarrow(\Box p_4 \wedge \Box\neg p_5) \tag{29}$$
$$\Leftrightarrow\Box(p_4 \wedge \Box\neg p_5) \tag{30}$$
$$\Rightarrow\Box(p_4 \wedge \Diamond r_5) \tag{31}$$
$$\Rightarrow\Box\Diamond(p_4 \wedge r_5) \tag{32}$$
$$\Rightarrow\Box\Diamond(ready[1] \wedge trun = 1) \tag{33}$$

Erklärung: Die erste Äquivalenz folgt daraus, dass wenn p_4 gilt, aber p_5 nie erreicht wird, dann muss auch immer p_4 gelten, da man keine Zeile im Code einfach überspringen darf. Die Zweite Äquivalenz ist aus Aufgabe 2 abzuleiten. Das impliziert das nächste Ausdruck, da es offensichtlich ist, dass T_0 nie in seinem CS eintritt, deswegen muss T_1 als erstes bis zu r_4 gelaufen sein und muss irgendwann mal in seine CS eintreten, also $\Diamond r_5$. Da aus diese Aussage zu lesen ist 'p_4 gilt immer UND immer wieder gilt r_5', dann ist es offensichtlich, dass immer wieder die beide zusammen gelten. Aus dem in den Klammern entstandenen Ausdruck mit Hilfe der 1. Invariante wird die letzte Implikation hergeleitet.

2) $turn \leftarrow 0$ wird initial gesetzt $\Rightarrow \Diamond(turn = 0)$. Da die Aussage eine Verknüpfung von $\Diamond(turn = 0)$ und eine andere Aussage ist, reicht nur das eine wahr zu sein um die ganze Aussage zu beweisen $\Rightarrow$ fertig.

Die andere Aussage ist unser Meinung nach nicht außer irgendwelche Sonderfälle zu beweisen, da dafür muss das NCS von T_0 nie beenden, damit nier ready[1] gesetzt wird. Sonsts ist es garantiert, dass wenn ready[1] auf True gesetzt wird und wieder auf False, dass das Loop von T_1 noch mindestens einen Durchlauf machen wird, in dem ready[1] wieder auf True gesetzt wird.

Alternativen Lösungsansatz:
Falls $\Diamond\Box(\neg ready[1])$ nicht wahr ist wird irgendwann r_3 ausgeführt (die Unwahrheit der 1. Aussage garantiert, dass NCS von T_1) irdgendwann zu Ende ausgeführt wird und das Programm läuft irgendwann weiter zu r_3 (falls das Scheduler fair ist, das soll aber für die Aufgabe nicht relevant sein).

Ist das eine Fangfrage?

3)

$$p_4 \wedge \Box\neg p_5 \wedge \Diamond(turn = 0) \tag{34}$$
$$\Rightarrow p_4 \wedge \Diamond r_5 \wedge \Diamond(turn = 0) \tag{35}$$
$$\Rightarrow\Diamond\Box(turn = 0) \tag{36}$$

Analog zu 1), $\Box\neg p_5$ bedeutet, dass T_0 als zweites turn zugewiesen hat und jetzt beim aktives warten ist. Deswegen muss T_1 irgendwann mindestens einmal sein CS ausgeführen. Wegen $\Diamond turn = 0$ können wir meinen, dass NCS von T_1 endlich ist und dass dieser nach seinem CS turn = 0 setzt und aktiv wartet. Wegen Scheduler kommt aber T_0 nicht dran (wegen $/negp_5$) und deswegen bleibt immer turn = 0, also $\Rightarrow \Diamond\Box(turn = 0)$.

Das heißt T_0 geht nie in die CS

5 Aufgabe

Damit die Threads nicht verhungern, muss den Zustand $(p_4 \wedge \Box\neg p_5) \wedge (r_4 \wedge \Box\neg r_5)$ nie vorkommen.

$(p_4 \wedge \Box\neg p_5) \wedge (r_4 \wedge \Box\neg r_5)$ (37)

$\Rightarrow T_0$ und T_1 sind beide zu 4. Schritt gekommen, also beides turn = 0 und turn = 1 wurden gesetzt (38)

$\Rightarrow \Diamond(turn = 0) \wedge \Diamond(turn = 1) \wedge (p_4 \wedge \Box\neg p_5) \wedge (r_4 \wedge \Box\neg r_5)$ (39)

$\Rightarrow$Von Aufgabe 4.3 kann man ableiten, dass (40)

$\Rightarrow \Diamond\Box(turn = 0) \wedge \Diamond(turn = 1) \wedge (r_4 \wedge \Box\neg r_5)$ (41)

$\Rightarrow \Diamond\Box(turn = 0) \wedge \Diamond\Box(turn = 1)$ (Analog für T_1)

$\Rightarrow$Widerspruch! Eine Variable kann nicht gleichzeitig 2 Werte haben! (42)

6 Aufgabe

T_p läuft bis $p_3 \Rightarrow T_p$::temp = 0
T_r läuft bis $r_3 \Rightarrow T_r$::temp = 0

T_p führt p_3 aus bzw. T_r führt r_3 aus. Jetzt sind ticket_p = ticket_r = 1

T_p und T_r führen gleichzeitig p_4 bzw. r_4 aus. Da ticket_p = ticket_r $\Rightarrow$ ticket_p $\leq$ ticket_r $\wedge$ ticket_r $\leq$ ticket_p.

$\Rightarrow$ Beide gehen in die CS ein! Wechselseitiger Ausschuss ist nicht gewährleistet!