Formale Grundlagen der Informatik III: Übung 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2014/2015
Wird in der Übungsgruppe vom 05. November bis 14. November behandelt

Lösung 1 Aussagenlogik

Welche der folgenden Formeln sind gültig, erfüllbar oder keines von beidem? Geben Sie eine erfüllende und nicht-erfüllende Interpretation der Variablen an.

a) $(p \leftrightarrow (r \vee q)) \rightarrow (r \rightarrow p)$

Lösung

p	q	r	A $r \vee q$	B $r \rightarrow p$	C $p \leftrightarrow A$	$C \rightarrow B$
0	0	0	0	1	1	1
0	0	1	1	0	0	1
0	1	0	1	1	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

$\Rightarrow$ Formel ist **gültig**.

b) $(p \wedge (q \rightarrow (r \vee p))) \rightarrow q$

Lösung

p	q	r	A $r \vee p$	B $q \rightarrow A$	C $p \wedge B$	$C \rightarrow q$
0	0	0	0	1	0	1
0	0	1	1	1	0	1
0	1	0	0	0	0	1
0	1	1	1	1	0	1
1	0	0	1	1	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	1	1	1	1

$\Rightarrow$ Formel ist **erfüllbar**.

c) $((p \wedge q) \wedge (\neg p \wedge (r \vee q))) \vee ((p \rightarrow \neg q) \wedge ((r \vee q) \rightarrow p))$

Lösung

p	**q**	**r**	**A** $\mathbf{r \vee p}$	**B** $\mathbf{\neg p \wedge A}$	**C** $\mathbf{p \wedge q}$	**D** $\mathbf{C \wedge B}$	**E** $\mathbf{A \rightarrow p}$	**F** $\mathbf{p \rightarrow \neg q}$	$\mathbf{F \wedge E}$
0	0	0	0	0	0	0	1	1	1
0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	0	1	0
0	1	1	1	1	0	0	0	1	0
1	0	0	0	0	0	0	1	1	1
1	0	1	1	0	0	0	1	1	1
1	1	0	1	0	0	0	1	0	0
1	1	1	1	0	1	0	1	0	0

⇒ Formel ist **erfüllbar**.

d) $((p \wedge r) \wedge ((q \rightarrow \neg p) \wedge (r \rightarrow q)))$

Lösung

p	**q**	**r**	**A** $\mathbf{p \wedge r}$	**B** $\mathbf{q \rightarrow \neg p}$	**C** $\mathbf{r \rightarrow q}$	**D** $\mathbf{B \wedge C}$	$\mathbf{A \wedge D}$
0	0	0	0	1	1	1	0
0	0	1	0	1	0	0	0
0	1	0	0	1	1	1	0
0	1	1	0	1	1	1	0
1	0	0	0	1	1	1	0
1	0	1	1	1	0	0	0
1	1	0	0	0	1	0	0
1	1	1	1	0	1	0	0

⇒ Formel ist **nicht erfüllbar**.

Lösung 2 CTL Spezifikation

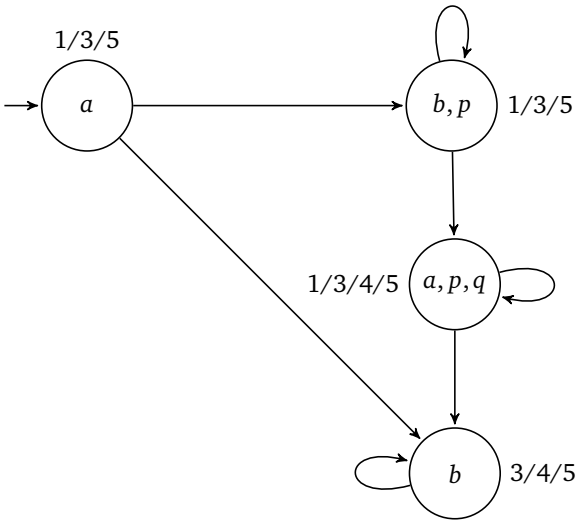

Abbildung 1: Kripke-Struktur zu Lösung 2

a) Markieren Sie in der Kripke-Struktur in Abbildung 3 all jene Zustände, in denen die folgenden Formeln gültig sind:

1. $\mathbf{EF}a$.
2. $\mathbf{AG}a$.
3. $\mathbf{E}a\mathbf{U}b$.

   Hinweis: a **U** b ist auch erfüllt, wenn ein b vorkommt ohne das vorherige Auftreten von a auf diesem Pfad.
4. $\mathbf{AG}(p \rightarrow q)$.
5. $(a \vee q) \rightarrow \mathbf{EX}b$.

Lösung

Siehe Markierungen in Abbildung 3.

b) Seien p, q atomare Eigenschaften des Systems. Formulieren Sie die folgenden Spezifikationen in CTL so einfach wie möglich: (Es können mehrere Lösungen möglich sein.)

Hinweis: **Durch die nicht immer eindeutigen textuellen Beschreibungen können Ihre Lösungen von unseren Lösungsvorschlägen abweichend sein. Dies gilt besonders für die Verwendung der Pfadquantoren E und A. Sollte eine vergleichbare Aufgabe in der Klausur vorkommen, würde dies klarer formuliert sein.**

1. p tritt niemals ein.

   Lösung

   $\mathbf{AG}\neg p$
2. p tritt mindestens zweimal in der Zukunft ein (zu zwei verschiedenen Zeitpunkten).

   Lösung

   $\mathbf{AF}(p \wedge \mathbf{AXAF}p)$
3. p tritt in den nächsten zwei Zeiteinheiten nicht ein.

   Lösung

   $\mathbf{AX}(\neg p \wedge \mathbf{AX}\neg p)$
4. Wann immer p eintritt, kann q nicht mehr eintreten.

   Lösung

   $\mathbf{AG}(p \rightarrow \mathbf{AG}\neg q)$
5. p gilt solange, bis p falsch wird.

   Lösung

   $\mathbf{A}p\mathbf{U}\neg p$
6. Entweder gilt p im nächsten Schritt oder es gilt nie mehr.

   Lösung

   $\mathbf{AX}(p \vee \mathbf{AG}\neg p)$
7. Wenn es möglich ist einen Zustand zu erreichen, in dem p gilt, so muss unendlich oft ein Zustand erreicht werden können, in dem p gilt.

   Lösung

   $\mathbf{AG}(p \rightarrow \mathbf{AGAF}p)$

c) Sind die folgenden Formeln erfüllbar, gültig oder keins von beidem? Falls die Formel erfüllbar ist, geben Sie eine Kripke-Struktur und einen Anfangszustand an, in der die Formel erfüllbar ist.

1. $(\mathbf{AG}p) \rightarrow (\mathbf{AG}\neg p)$.

   Lösung

   $\Rightarrow$ Formel ist **erfüllbar**.

2. $(\mathbf{AG}p) \rightarrow (\mathbf{AG}p)$.

   Lösung

   $\Rightarrow$ Formel ist **gültig**.

3. $(\mathbf{AF}p) \rightarrow (\mathbf{EF}p)$.

   Lösung

   $\Rightarrow$ Formel ist **gültig**.

4. $(p \wedge \neg p) \leftarrow (q \wedge \neg p)$.

   Lösung

   $\Rightarrow$ Formel ist **erfüllbar**.

Abbildung 2: Kripke-Struktur zu 4.

5. $(p \wedge \neg p) \rightarrow \text{false}$.

   Lösung

   $\Rightarrow$ Formel ist **gültig**.

6. $(\mathbf{AX}p) \rightarrow (\mathbf{EF}p)$.

   Lösung

   $\Rightarrow$ Formel ist **gültig**.

7. $(\mathbf{AX}p) \rightarrow (\mathbf{EF}\neg p)$.

   Lösung

   $\Rightarrow$ Formel ist **erfüllbar**.

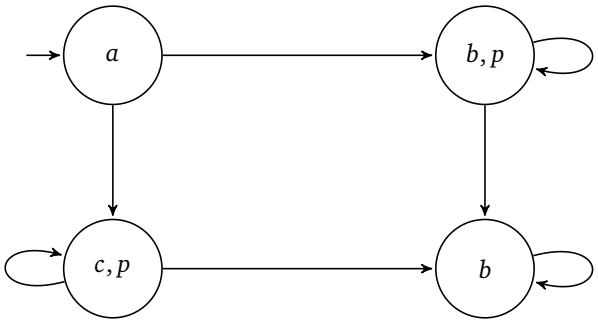

Abbildung 3: Kripke-Struktur zu 7.

d) Stellen Sie die folgenden CTL Formeln nur durch **EX**, **EU**, **EG** dar:

Anmerkung: $\mathbf{AF}\varphi = \neg\mathbf{EG}\neg\varphi$

1. $\mathbf{EF}(s \wedge \neg r)$

Lösung

$$\mathbf{EF}(s \wedge \neg r) = \mathbf{E}\top\mathbf{U}(s \wedge \neg r) \tag{1}$$

2. $\mathbf{AG}(r \rightarrow \mathbf{AF}ack)$

Lösung

$$\begin{aligned}\mathbf{AG}(r \rightarrow \mathbf{AF}ack) &= \mathbf{AG}(r \rightarrow \neg\mathbf{EG}\neg ack)\\ &= \neg\mathbf{EF}\neg(r \rightarrow \neg\mathbf{EG}\neg ack)\\ &= \neg\mathbf{E}\top\mathbf{U}\neg(r \rightarrow \neg\mathbf{EG}\neg ack)\end{aligned} \tag{2}$$

3. $\mathbf{AGAF}e$

Lösung

$$\begin{aligned}\mathbf{AGAF}e &= \mathbf{AG}\neg\mathbf{EG}\neg e\\ &= \neg\mathbf{EF}\neg(\neg\mathbf{EG}\neg e)\\ &= \neg\mathbf{EFEG}e\\ &= \neg\mathbf{E}\top\mathbf{UEG}e\end{aligned} \tag{3}$$

4. $\mathbf{AGEF}r$

Lösung

$$\begin{aligned}\mathbf{AGEF}r &= \mathbf{AGE}\top\mathbf{U}r\\ &= \neg\mathbf{EF}(\neg\mathbf{E}\top\mathbf{U}r)\\ &= \neg\mathbf{E}\top\mathbf{U}(\neg\mathbf{E}\top\mathbf{U}r)\end{aligned} \tag{4}$$

Lösung 3 Experimentieren mit SPIN und PROMELA

In Abbildung 4 wird die Belegung eines Bahnhofsgleises in Form eines Petri-Netzes dargestellt.

Hintergrundwissen: *Petri-Netze stellen Modelle diskreter Systeme dar. Sie gehen von endlichen Automaten aus. In der vorliegenden Notation repräsentieren die Kreise jeweils einen Zustand des Systems. Die Punkte (Token) innerhalb von jedem Zustand werden für das Durchführen von Transitionen benötigt. Transitionen werden durch die Quadrate dargestellt. Zur Durchführung einer Transition muss in jedem Zustand, der eine Kante in Richtung der Transition haben ein Token vorliegen. Nach Durchführung der Transition wird aus diesen Zuständen je ein Token entfernt und ein die Zustände verschoben, zu denen die Transition führt.*

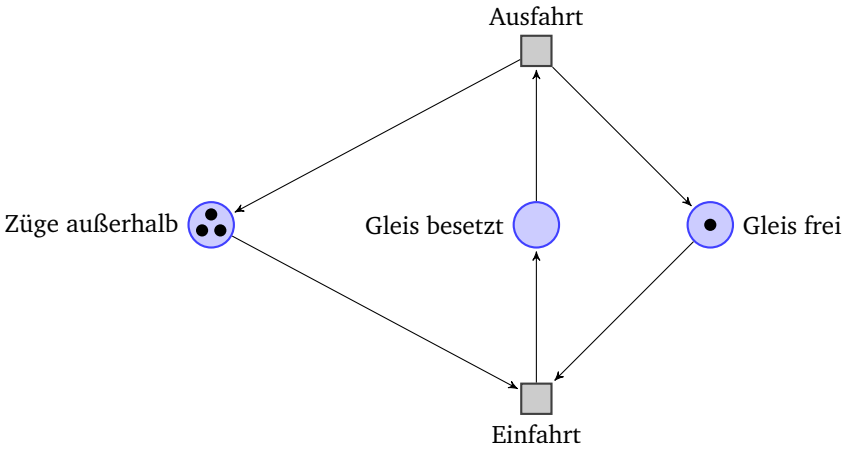

Abbildung 4: Belegung eines Bahnhofsgleises

a) Erstellen Sie ein PROMELA Model des in Abbildung 4 gegebenen Petri-Netzes.

Lösung

```
#define State byte

State za,gb,gf;

#define trans_in1(x)     (x>0) -> x=x-1
#define trans_in2(x,y)   (x>0 && y>0) -> x=x-1;y=y-1
#define trans_out1(x)    x=x+1
#define trans_out2(x,y)  x=x+1; y=y+1

proctype Station(){
  za = 3; gf = 1;
  do
  :: atomic{trans_in2(za, gf) -> trans_out1(gb)}
  :: atomic{trans_in1(gb) -> trans_out2(za,gf)}
  od
}

init{
  run Station();
}
```

Alternative ohne die Verwendung von Makros:

```
#define State byte

State za,gb,gf;
```

```
proctype Station(){
  za = 3; gf = 1;
  do
  :: atomic{(za > 0 && gf >0) -> za = za-1; gf = gf-1; gb = gb+1}
  :: atomic{(gb > 0) -> gb = gb-1; za = za+1; gf = gf+1}
  od
}

init{
  run Station();
}
```

b) Führen Sie eine Simulation mit 100 Schritten durch. Wie viele erreichbare Zustände werden durch das PROMELA Model generiert?

Lösung

```
spin -p -u100 ex1.pml
```

Die Zustände können anschließend aus dem Simulations-Trace abgelesen werden.

c) Ist in dem Netz ein Deadlock enthalten? Verifiziere dies mittels SPIN.

Lösung

Dies kann durch die Erstellung eines Verifikationsmodells geprüft werden:

```
spin -a a3.pml
gcc -o pan pan.c
./pan -d
```

Falls ein Deadlock vorhanden ist wird von dem Verifizierer ein illegaler Endzustand zurückgegeben. Dies könnte zum Beispiel wie folgt aussehen:

```
def@def-debian:~/spin/ex1$ ./pan
hint: this search is more efficient if pan.c is compiled -DSAFETY
pan:1: invalid end state (at depth 2)
pan: wrote a3.pml.trail

(Spin Version 6.4.2 -- 8 October 2014)
Warning: Search not completed
        + Partial Order Reduction
```

Den entsprechenden Fehler kann man sich anschließend über die generierte Trail Datei ansehen:

```
def@def-debian:~/spin/ex1$ spin -t a3.pml
spin: trail ends after 3 steps
#processes: 2
        za = 3
        gb = 0
        gf = 1
3:      proc  1 (Station:1) a3.pml:12 (state 5)
3:      proc  0 (:init::1) a3.pml:23 (state 2) <valid end state>
2 processes created
```

Im Fall des obigen Codes ist allerdings kein Deadlock vorhanden, daher erhalten wir keine entsprechende Fehlermeldung. Der Verifizierer kann seine Arbeit ganz normal abschließen.