

FGI-2 – Formale Grundlagen der Informatik II

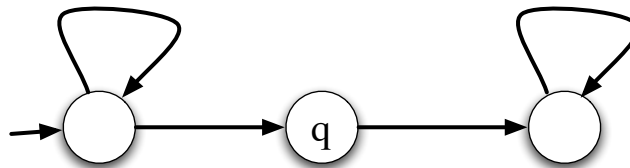
Prozesse und Nebenläufigkeit

Aufgabenblatt 9: Analyse von Systemen

Abgabe am 8.1.2007 Besprechung am 10.1.2007.

Präsenzaufgabe 9: CTL

Hans sagt: „ $AGAFp$ heißt: Auf allen Abläufen gilt unendlich oft p . Also wird $EGEFq$ heißen: Es gibt einen Ablauf, auf dem unendlich oft q gilt.“ Lisa bestreitet dies mit folgender Kripke-Struktur. Wer hat recht?



Übungsaufgabe 9.1:

Betrachten Sie den bekannten **Algorithmus von Peterson/Dekker für wechselseitigen Ausschluss** in Abbildung 3!

- Diskutieren Sie konkret (d.h. mit Bezug auf die entsprechenden Programmzeilen), ob der wechselseitige Ausschluss eingehalten wird und warum ein Prozess, der den Eintritt in den kritischen Abschnitt wünscht (was heißt das?), irgendwann einmal den Zutritt schafft?
- Gehören die folgenden Formeln $AG(\neg(p_4 \wedge q_4))$ und $AG(\neg p_1 \vee AFp_4)$ zu *CTL*? Was bedeuten sie und gelten sie für das Programm?
- Bei der Konstruktion der zugehörigen Kripke-Struktur ist es wichtig, möglichst wenig Zustände zu erzeugen. Warum genügt es, um obige Formeln zu prüfen, die Zeilen p_0, q_0, p_4, q_4 zu streichen und dann die Formeln $AG(\neg(p_5 \wedge q_5))$ und $AG(\neg p_1 \vee AFp_5)$ zu überprüfen?
- Prüfen Sie, ob Abbildung 4 die Kripke-Struktur für das reduzierte Programm darstellt! Dabei soll eine Knotenbeschriftung (i, j, n, b_1, b_2) den Zustand $(p_i, q_j, last = n, wantP = b_1, wantQ = b_2)$ bezeichnen. Prüfen Sie hier zunächst ohne Algorithmus, ob die in c) genannten Spezifikationen gelten.
- Wenden Sie nun den *CTL*-Algorithmus auf diese Formeln an. Geben Sie zunächst die nötigen Umformungen an, wobei wie im Beispiel des Skriptes \wedge und *EF* stehen bleiben dürfen, wenn dadurch die Formel einfacher wird. Beschreiben Sie den Ablauf des Algorithmus, in dem Sie jeweils wie im Skript für jeden Schritt die Zustandsmengen $S(\text{Teilformel}) = \{s_x, \dots\}$ angeben!
- Was ändert sich in der durchgeführten Untersuchung, wenn die faire Kripke-Struktur mit $F = \{P_1, P_2\}$ vorausgesetzt wird. Dabei sei $P_1 = \{p_1, q_1, q_2, q_3, q_5\}$ und $P_2 = \{q_1, p_1, p_2, p_3, p_5\}$.

VON
10

Bisher erreichbare Punktzahl:

95

```

wantP = wantQ = False : boolean,
last = 1 : integer,
cobegin P || Q coend
P::
while True do
  [p0 : non-critical section;]
  p1 : wantP := True;
  p2 : last := 1;
  p3 : wait(wantQ = False ∨ last = 2);
  [p4 : critical section;]
  p5 : wantP := False;
endwhile;

Q::
while True do
  [q0 : non-critical section;]
  q1 : wantQ := True;
  q2 : last := 2;
  q3 : wait(wantP = False ∨ last = 1);
  [q4 : critical section;]
  q5 : wantQ := False;
endwhile;

```

Abbildung 3: Algorithmus von Peterson/Dekker für wechselseitigen Ausschluss

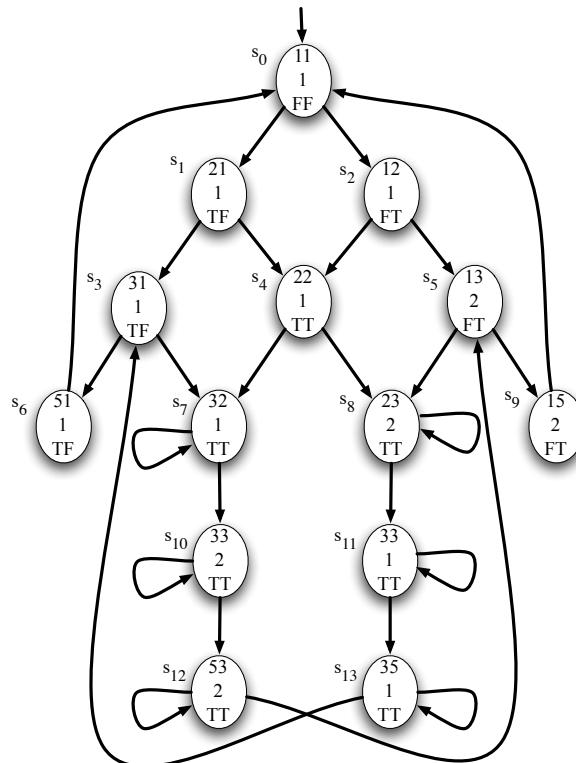


Abbildung 4: Kripke-Struktur