Dr. Ulrich Schöpp

Ludwig-Maximilians-Universität München
Institut für Informatik
SS 2016

Übungen zur Vorlesung
Formale Spezifikation und Verifikation

Blatt 5

Aufgabe 5-1 Überprüfen Sie die Antworten für Aufgabe 4-2 mit NuSMV.

Hinweis: Im CIP-Pool ist eine ältere Version von NuSMV installiert, die auf der Kommandozeile direkt mit „`NuSMV datei.smv`“ aufgerufen werden kann.

Die neueste NuSMV-Version kann mit „`/home/sch/schoepp/bin/NuSMV datei.smv`“ ausgeführt werden.

Aufgabe 5-2 Wir betrachten die Situation, in der zwei Java-Threads parallel ablaufen und dabei beide den folgenden Code ausführen.

```
x = 1;
while (true) {
  x = x + x;
}
```

Darin ist `x` eine gemeinsame Variable vom Typ `int`. Wir untersuchen, welche Werte diese gemeinsame Variable `x` im Laufe der Ausführung annehmen kann.

Auf Bytecode-Ebene werden Anweisungen wie `x = x + x;` nicht in einem Schritt ausgeführt. Für diese Anweisung werden erst beide Summanden in temporäre Register geladen, dann wird die Summe berechnet und dann wird das Ergebnis in `x` geschrieben. Der Bytecode für obigen Java-Code entspricht folgendem Pseudo-Code:

```
1:  x := 1; goto 2;
2:  u := x; goto 3;
3:  v := x; goto 4;
4:  w := u + v; goto 5;
5:  x := w; goto 2;
```

Darin sind `u`, `v` und `w` temporäre `int`-Variablen.

Ihre Aufgabe ist nun, mit NuSMV zu untersuchen, welche Werte die Variable `x` annehmen kann. Beschränken Sie dabei die möglichen Werte der `int`-Variablen zunächst auf eine sehr kleine Menge wie $\{1, \ldots, 16\}$ anstelle von $\{-2^{32}, \ldots, 2^{32}\}$.

a) Kodieren Sie die parallele Ausführung zweier Prozesse mit obigem Pseudocode in NuSMV. Sie können sich am Semaphoren-Beispiel aus der Vorlesung orientieren. Verwenden Sie für `int`-Werte den NuSMV-Typ `1..16` (d.h. Zahlen von 1 bis 16). NuSMV unterstützt Operationen wie `x+y`, `max(x,y)` und `min(x,y)` für solche Zahlwerte.

b) Untersuchen Sie, welche Werte aus $\{1, \ldots, 16\}$ die Variable `x` im Laufe der Ausführung der beiden Prozesse annehmen kann. Lassen Sie sich für jeden Wert, den `x` annehmen kann, von NuSMV einen entsprechenden Lauf des Systems berechnen.

c) Können Sie Ihre Ergebnisse verallgemeinern?

Aufgabe 5-3 Um sich die Spezifikation von Systemeigenschaften in CTL zu erleichtern, kann man sich für häufig auftretende Spezifikationsmuster CTL-Formeln bereitlegen, aus denen man dann die gewünschte Formel aufbaut. Im Folgenden sind einige Eigenschaften angegeben, die als solche Spezifikationsmuster vorgeschlagen wurden.

Drücken Sie folgende Systemeigenschaften in CTL aus. Die Pfade im Transitionssystem sind die möglichen Systemausführungen.

a) Wenn q in einer Ausführung des Systems irgendwann wahr wird, so wird später auch p wahr.

b) Wenn q in einer Ausführung des Systems irgendwann wahr wird, dann ist p die ganze Zeit bis dahin wahr.

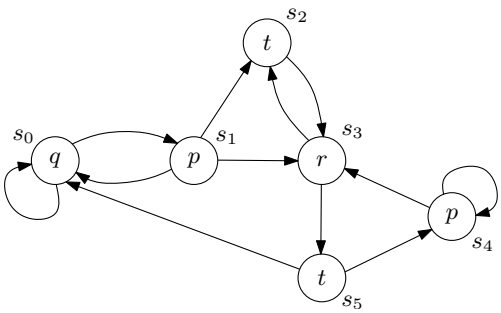
c) Wenn q in einer Ausführung irgendwann wahr wird, dann muss p vorher mindestens einmal wahr gewesen sein.

d) Wenn q irgendwann wahr wird, so wird später auch p wahr, und zwar noch bevor f wahr wird.

e) Wenn q irgendwann nach e wahr wird, so wird später auch p wahr.

In allen Fällen sind die Zeitspannen jeweils inklusive der Endpunkte zu verstehen.

Aufgabe 5-4 Gegeben sei ein Transitionssystem mit einer Interpretation der aussagenlogischen Variablen $\{p, q, r, t\}$ wie folgt:



Sei φ die Formel $\mathrm{EG}\,(\mathrm{E}[(t \vee r)\,\mathrm{U}\,q])$.

a) Wenden Sie die effizientere Variante des Labelling-Algorithmus, in der EG direkt behandelt wird, auf die Formel φ an und entscheiden Sie so, an welchen Zuständen diese gilt.

b) Wenden Sie den analogen Algorithmus zum Model-Checking von CTL mit Fairness mit den beiden durch t und q gegebenen Fairnessbedingungen, d.h. es sind die Fairnessbedingungen (f_1, f_2) mit $f_1 = \{s_2, s_5\}$ und $f_2 = \{s_0\}$ gegeben.

Geben Sie bei allen Punkten die starken Zusammenhangskomponenten an, die bei der Anwendung des Algorithmus vorkommen und kennzeichnen Sie, welche davon fair sind.

Abgabe: Sie können Ihre Lösungen bis Mittwoch, den 1.6., um 16:00 Uhr über UniWorX abgeben.