Gruppe: _____ Übung 6

Name: _____ **Systemtheorie 1**

Matr.Nr.: _____ Wintersemester 2006/2007

Erfolg: _____ Abgabe 07.12.2006 10:30

Institut für Formale Modelle und Verifikation, Dr. Toni Jussila, Dipl.-Ing. Robert Brummayer

Aufgabe 21

a) Finden Sie ein Beispiel für die erste fehlerhafte Variante des DFS-Algorithmus auf Folie 14 in Folienset *reach*, bei welchem der Algorithmus nicht korrekt funktioniert. Simulieren Sie den Algorithmus schrittweise auf diesem Beispiel. Erklären Sie schließlich, warum der Algorithmus nicht fehlerfrei arbeitet.

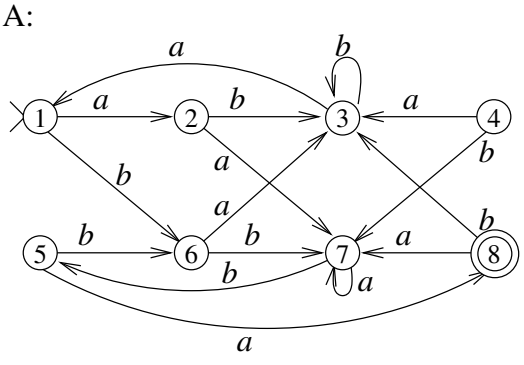
b) Führen Sie den gleichen Vorgang in *a*) mit der zweiten statt der ersten fehlerhaften Variante des DFS-Algorithmus auf Folie 15 durch.

Verwenden Sie jeweils folgende Datenstrukturen:

- *Cache* ist die Menge aller bereits besuchten und somit markierten Zustände. Zu Beginn ist diese Menge leer.
- *Stack* ist der Stack, welche jene Zustände enthält, die der Algorithmus als nächstes besuchen soll. Dieser Stack wird mit den Startzuständen initialisiert.

Aufgabe 22

A:



Wenden Sie den BFS-Algorithmus auf Folie 16 und 17 in Folienset *reach* schrittweise manuell auf *A* an. Besuchen Sie dabei immer jeweils zuerst den Folgezustand, der mit *b* erreichbar ist. Verwenden Sie folgende Datenstrukturen:

- *Cache* ist die Menge aller bereits besuchten und somit markierten Zustände. Zu Beginn ist diese Menge leer.
- *Queue* ist die Warteschlange, welche jene Zustände enthält, die als nächstes besucht werden sollen. Diese Warteschlange wird mit den Startzuständen initialisiert.

Sobald der Finalzustand erreicht wird, können Sie mit der Simulation aufhören. Geben Sie schließlich die vom BFS-Algorithmus gefundene Folge von Zuständen, die vom Anfangs- zum Endzustand führt, an.

Aufgabe 23

Für diese und zukünftige Aufgaben benötigen Sie einen C-Compiler. Falls auf ihrem System kein C-Compiler verfügbar ist, installieren Sie einen. Freie C-Compiler für Windows sind beispielsweise:

- GCC in Cygwin: www.cygwin.com
- GCC in MinGW: www.mingw.org
- lcc-win32: http://www.cs.virginia.edu/ lcc-win32/

Unter Linux und Mac OS X ist GCC zu empfehlen.

a) Übersetzen Sie das C-Programm auf Folie 11 in Folienset *reach* und lassen Sie es schließlich laufen. Bei welcher Rekursionstiefe stürzt das Programm auf ihrem System ab? Geben Sie einen Screenshot vom letzten Teil der Ausgabe ab.

b) Schätzen Sie aufgrund von *a*) wie groß der Stack-Speicher ist, den ihr Compiler anlegt.

Hinweis: Verwenden Sie beim Übersetzen des Programmes keine Compiler-Optimierungen.

Aufgabe 24

Experimentieren Sie mit dem Open-Source Model-Checker für verteilte Software-Systeme: `SPIN`. Laden Sie `SPIN` unter http://spinroot.com herunter. Es gibt vorkompilierte Executables für Windows, 32-Bit- und 64-Bit Linux. Wer möchte, kann `SPIN` auch direkt aus den Source-Dateien übersetzen. `SPIN` ist für Windows, Linux und Mac OS frei verfügbar.

Gleich neben der Übungsangabe finden Sie eine Eingabe-Datei `exercise24.pml`. Diese Datei enthält einen verteilten Algorithmus zur Berechnung eines Hash-Wertes einer Matrikerlnummer. Die Matrikelnummer wird in einem Byte-Array der Länge 7 gespeichert. Modifizieren Sie diese Datei, indem Sie die vorgegebene Matrikelnummer durch ihre Matrikelnummer im Init-Prozess ersetzen. Lassen Sie schließlich `SPIN` auf der Kommandozeile ohne Argumente mit der Eingabe-Datei laufen. Geben Sie einen Screenshot von der Ausgabe von `SPIN` ab.

Hinweis: Es wird noch weitere Übungsbeispiele zu `SPIN` geben.