Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Frank Huch

24118 Kiel
Christian-Albrechts-Platz 4
Tel.: 0431 / 880-7277

10. Übung „Nebenläufige und verteilte Programmierung“

Abgabe in der Vorlesung oder bis zum 28. Januar in Raum 704

Aufgabe 24 **6 Punkte**

Machen Sie den Chat aus Aufgabe 23 robust. Insbesondere soll der Chat-Server terminierte Chat-Room-Server neu starten, ohne daß die Chat-Klienten dies bemerken. Das ist natürlich nur solange möglich, bis keine Knoten mehr für den Start von Chat-Room-Servern zur Verfügung stehen.

Aufgabe 25 **8 Punkte**

In dieser Aufgabe sollen Sie die Ausdrucksstärke von nebenläufigen Systemen untersuchen. Ziel ist es Simulationen von Turing-Maschinen zu implementieren.

a) Implementieren Sie eine Turing-Maschine mit Hilfe eines endrekursiven Erlangprogramms (nach einem Funktionsaufruf darf keine weitere Berechnung mehr erfolgen). Ihr Implementierung soll nur eine endliche Anzahl von Atomen verwenden. Repräsentieren Sie das Band der Turing-Maschine durch nebenläufige Server-Prozesse, welche den aktuellen Buchstaben und die Pids ihrer Nachbarprozesse als Zustand halten. Ein zusätzlicher Kontrollprozeß stellt die endliche Kontrolle der Turingmaschine dar, welche mit dem Prozeß der aktuellen Schreib-/Leseposition kommuniziert. Ihre Implementierung darf keine Listen oder verschachtelte Tupel verwenden.

   Eine Turing-Maschine $\mathcal{A} = \langle Q, \Gamma, \delta, q_0, F\rangle$ sein durch folgende Erlang-Funktionen gegeben:

   `delta(`q`,`a`) -> {`p`,`b`,`r`}` `falls` $\delta(q,a) = (p,b,r)$ `mit` $q,p \in Q$, $a,b \in \Gamma$ `und` $r \in \{$`r,l`$\}$

   `final(`q`) -> true` `falls` $q \in F$

   `final(`q`) -> false` `falls` $q \notin F$

   Beispiel: Erkennung der Sprache $\{a^n b^n \mid n \geq 0\}$:

```
delta(q0,a) -> {q1,blank,r};
delta(q0,blank) -> {f,blank};
delta(q1,a) -> {q1,a,r};
delta(q1,b) -> {q2,b,r};
delta(q2,blank) -> {q3,blank,l};
delta(q3,b) -> {q4,blank,l};
delta(q4,b) -> {q4,b,l};
```

```
delta(q4,a) -> {q4,a,l};
delta(q4,blank) -> {q0,blank,r).

final(f) = true;
final(_) = false.
```

b) Schreiben Sie ein Erlang-Programm, welches mit nur drei Prozessen eine Turing-Maschine simuliert. Diese Variante soll neben den Buchstaben des Alphabets Γ maximal sieben Werte (Atome und Pids verwenden).

Definieren Sie zunächst mit Hilfe nicht-endrekursiver Funktionsaufrufe einen Stack. Erweitern Sie den Stack so, daß er beim Lesen des leeren Stacks zusätzliche `blank`s liefert. Mit Hilfe von zwei Stack-Prozessen und einem Kontrollprozeß können Sie dann eine Turingmaschine definieren.

c) Welche Konsequenzen für die formale Verifikation nebenläufiger Systeme haben Ihre Simulationen?

Aufgabe 26 **6 Punkte**

Bei der Programmierung verteilter Spiele ist die initiale Phase des Zusammenkommens der Spielpartner ein häufig wiederkehrendes Problem. Definieren Sie ein Erlang-Programm, welches das Zusammentreffen mehrerer Spieler ermöglicht. Ein Spieler soll ein Spiel starten können, welchem sich andere Spieler anschließen können. Das Hinzukommen neuer Spieler soll jedem Spieler angezeigt werden. Neu hinzugekommene Spieler sollen ebenfalls alle bereits vorhandenen Mitspieler angezeigt bekommen. Das Spiel soll starten, wenn ein beliebiger Spieler den Start des Spiels frei gibt. Zu diesem Zeitpunkt soll allen Spielern eine Liste der Pids aller Mitspieler und die ausgezeichnete Pid des initiierenden Spielers zur Verfügung stehen.
Erweitern Sie Ihr Erlang-Programm zu einem Modul, welches von beliebigen Spielen verwendet werden kann. Wie können Sie elegant das konkrete Spiel in Ihr Modul einbinden?
Wer Lust hat kann dieses Modul auch noch robust gestalten, so daß zwischendurch Spieler auch wieder terminieren/abstürzen können.