Lösungsvorschläge für das 4. Übungsblatt zur Vorlesung „FMiSE“

TECHNISCHE UNIVERSITÄT DARMSTADT

Modellierung verteilter Systeme (Message Passing/Channels)
Diskussion der Aufgaben und Lösungen in den Tutorien: 21.5. – 24.5.2024

1 Verteilte Systeme: Grundlegendes

Channels (dt. Kanäle) sind nützliche Konzepte für die Modellierung verteilter Systeme. Bei der Deklaration eines PROMELA Channels wird dessen Kapazität und Nachrichtentyp festgelegt.

Im Beispiel

```
chan myChan = [0] of { byte }

active proctype producer() {
    // ...
}

active proctype consumer() {
    // ...
}
```

wird ein Channel namens myChan der Kapazität 0 (also einen Rendez-Vous Channel) mit Nachrichtentyp **byte** deklariert.

Modellieren Sie die zwei Prozesse producer und consumer mit folgendem Verhalten:

1. Prozess producer sendet die Zahlen von 0 bis 9 über den Kanal.
2. Prozess consumer empfängt die Zahlen und gibt sie auf der Konsole aus.

Lösungsvorschlag:

Lösung:

Siehe Datei `solutions/exercise1.pml`.
Bei den Traces der Simulationsläufe mit Spin lässt sich gut erkenen, dass bei Rendez-Vous Channels das Senden und Empfangen der Nachricht im selben Schritt geschieht, d.h. die Befehle myChan ! i und myChan ? i werden gleichzeitig ausgeführt.

2 Drucken im Büro (Revisited)

Modellieren Sie analog zum vorherigen Übungsblatt ein Büro bei dem sich zwei Rechner einen Drucker teilen. Beide Rechner dürfen jedoch nicht gleichzeitig drucken.

Organisieren Sie den Zugriff auf den Drucker mit Hilfe eines Rendezvous Channels und verifizieren Sie, dass ihr Modell Mutual Exclusion sicher stellt.

Lösungsvorschlag:

Lösung:

Siehe Datei `solutions/printer.pml`

3 Channels mit Buffer

Erweitern Sie das PROMELA-Modell aus Aufgabe 1 wie folgt:

1. Es gibt nun 3 producer- und 2 consumer-Prozesse.
2. Jeder producer-Prozess sendet neben der Zahl auch einen privaten Antwort-Kanal (replyChannel) in den request-Kanal.
3. Nach der Ausgabe der empfangen Zahl, antwortet der consumer-Prozess über den privaten Kanal. Die Nachricht enhält nur die Prozess-Id des consumers.
4. Der request-Kanal ist buffered mit Kapazität 2.

Eine Vorlage finden sie in `files/exercise3.pml`.

Lösungsvorschlag:

Lösung:

Siehe Datei `solutions/exercise3.pml`.
Setzen Sie in iSpin „Physical Memory Available“ auf 10240 und „Maximum Search Depth“ auf 100000000.
Mit der Komandozeile: `spin -run -m100000000 -DMEMLIMIT=10240 exercise3.pml`

4 Alternating Bit Protokoll

In dieser Aufgabe modellieren Sie ein einfaches Sliding Window Protokoll (mit Window Größe 1), das sogenannte *Alternating Bit Protocol*.

Gegeben seien ein Sender und ein Empfänger, die untereinander Nachrichten über zwei Channels (request und response) austauschen.

Der Sender schickt die Nachricht msg zusammen mit einer 1-bit Sequenznummer (also 0 oder 1) über den request Channel an den Empfänger. Nach Erhalt der Nachricht, schickt der Empfänger ein Acknowledgment ack mit der Sequenznummer aus der empfangenen Nachricht über den response Channel an den Sender zurück.

Der Sender vergleicht die enthaltene Sequenznummer mit der der geschickten Nachricht, (i) stimmen beide überein, so wechselt er die Sequenznummer zum Komplement (also von 0 zu 1 bzw. von 1 zu 0) und schickt die nächste Nachricht an den Empfänger dem gleichen Protokoll folgend; (ii) sind beide Nummern unterschiedlich, verwirft der Sender das Acknowledgement und wartet weiter.

1. Implementieren Sie das oben beschriebene Protokoll in PROMELA.

2. Für das beschriebene Szenario ist das angegebene Protokoll unnötig kompliziert. Die Sequenznummer wird nicht wirklich benötigt, da PROMELA Channels perfekt sind (keine Nachrichten verlieren) und die nächste Nachricht erst nach erfolgreicher Bestätigung der vorhergehenden versandt wird. Ändern Sie das PROMELA Modell nun so ab, dass Nachrichten verloren gehen können. Des weiteren gilt folgendes, erhält der Sender für eine gewisse Zeit keine Nachricht, so wird ein Timeout ausgelöst und der Sender verschickt die Nachricht ein weiteres Mal mit der gleichen Sequenznummer wie vorher. Überlegen Sie sich dazu, wie Sie dieses Verhalten mit PROMELA modellieren können, obwohl die eigentlichen Channels perfekt sind und keine Nachrichten verlieren.
   Hinweis: Die Boolsche globale und nur lesbare Variable **timeout** hat den Wert **true**, falls kein laufender Prozess mehr eine Zuweisung ausführen kann. Ansonsten hast sie den Wert **false**. Damit lassen sich zwar keine festen Zeiträume definieren, jedoch erlaubt es diese Variable das Verhalten eines Systems im Falle eines Timeouts zu modellieren.

Lösungsvorschlag:

Lösung:

1. Siehe Datei `solutions/slidingWindowProtocol`.
   Die Channeltypen sind jeweils das Tupel mytype$\times$ **bit**.
   Somit wird das Tupel (msg, seqNr) gesendet bzw. (ack, ackNr) empfangen.
2. Siehe Datei `solutions/slidingWindowProtocol2`.
   Nachrichten können nun an den zwei markierten Stellen verloren gehen.