14. November 2002

Fachbereich Informatik
Lehrstuhl Informatik IV
Peter Herrmann, Veye Tatah
(herrmann@ls4.cs.uni-dortmund.de,
tatah@ls4.cs.uni-dortmund.de)

# 5. Übungsblatt RvS

1. **Aufgabe:** (3 Punkte)
   Auf der Rückseite finden Sie die Promela-Spezifikation eines einfachen verbindungsorientierten Kommunikationsdienstes. Dabei werden zwei Dienstzugangspunkte $A$ und $B$ durch gleichnamige Promela-Prozesse modelliert, die miteinander über zwei Kanäle *ab* bzw. *ba* kommunizieren. Der Systemzustand des Dienstes wird durch die Variable *state* angegeben.

   a) Welche Dienstleistungen bietet dieser Dienst? Wie sieht die Rechtevergabe bezüglich des Verbindungsabbaus aus?
   b) Geben Sie den Dienst als Mealy-Automat an.

   *Hinweis:* Zum Verständnis hilft es sehr, den Dienst durch Xspin simulieren und als Weg-/Zeitdiagramm anzeigen zu lassen. Die Spezifikation kann von unserer Übungsblattseite `http://ls4-www.cs.tu-dortmund.de/~tatah/rvs.html` geladen werden.

2. **Aufgabe:** (3 Punkte)
   Ergänzen Sie die Promela-Spezifikation derartig, dass die Verbindung von beiden Seiten aus abgebaut werden kann.

3. **Aufgabe:** (4 Punkte)
   Führen Sie nun eine Vollduplex-Datenübertragung ein. Verwenden Sie dazu die in der Spezifikation angegebenen Kanäle *abdata* und *badata*, über die Sie Nachrichten vom Typ *Data* schicken können. Dateneinheiten, die nach dem Verbindungsabbau eintreffen, sollen auch aus dem Kanal gelesen werden. Jedoch soll hier im Weg-/Zeitdiagramm statt eines *DataInd* eine Fehlermeldung angezeigt werden.

   *Hinweis:* Sie können für die Aufgaben 2. und 3. einen gemeinsamen Spezifikations-Ausdruck abgeben.

**Abgabe:** Donnerstags im Pavillon 6, Briefkästen 50, 51 und 52
bis spätestens 12:00 Uhr
**Besprechung:** in den Übungen der darauffolgenden Woche

```
♯define idle        0
♯define wfconind    1
♯define conind      2
♯define wfconcnf    3
♯define connected   4
♯define wfdisind    5

mtype = { Con, ConCf, Dis, Data };
byte state = idle;

chan ab = [10] of { mtype };
chan ba = [10] of { mtype };
chan abdata = [10] of { mtype };
chan badata = [10] of { mtype };

proctype A () {
    do
        :: atomic{ state == idle ->
            state = wfconind;
            printf("MSC: ConReq\n");
            ab!Con; }
        :: atomic{ ba?ConCf ->
            printf("MSC: ConCnf\n");
            state = connected; }
        :: atomic{ state == connected ->
            state = wfdisind;
            printf("MSC: DisReq\n");
            ab!Dis; }
    od
}

proctype B () {
    do
        :: atomic{ ab?Con ->
            printf("MSC: ConInd\n");
            state = conind; }
        :: atomic{ state == conind ->
            state = wfconcnf;
            printf("MSC: ConRes\n");
            ba!ConCf; }
        :: atomic{ ab?Dis ->
            printf("MSC: DisInd\n");
            state = idle; }
    od
}

init {
    atomic {
        run A ();
        run B (); }
}
```