Informatik – Übungsaufgaben

Steuerung von Prozessen

Tobias Krähling
eMail: <Tobias.Kraehling@SemiByte.de>
Homepage: <www.SemiByte.de>

12.02.2007
Version: 1.0

Zusammenfassung

Die Übungsaufgaben stammen aus den Übungsaufgaben und Anwesenheitsaufgaben zur Vorlesung »Einführung in die Informatik« von Herrn Prof. Dr. Wassermann an der Ruhr-Universität Bochum im Wintersemester 2006/07.

In diesem Dokument sind sowohl die Übungsaufgaben wie auch Lösungsvorschläge enthalten.

Inhaltsverzeichnis

1. Übungsaufgaben

▹ Frage 1

Entwerfen Sie einen Fahrkartenautomaten für ein Nahverkehrsunternehmen. Es gibt zwei Fahrpreisstufen: Ticket A für 2 € oder Ticket B für 3 €. Der Automat hat eine Wahltaste für jede Fahrpreisstufe und zwei weitere Wahltasten, ob man für einen Aufpreis von 1 € die Karte für die Benutzung in der Stoßzeit vor 9 Uhr morgens validieren möchte.

Vor dem Einwurf des Geldes ist zuerst das gewünschte Ticket zu wählen, anschließend kann man eventuell die Gültigkeit vor 9 Uhr hinzubuchen. Danach kann dann das Geld eingeworfen werden. Die Maschine nimmt nur 1 €- und 2 €-Münzen an.

Hat der Kunde genügend Geld eingeworfen, erhält er die gewählte Fahrkarte und eventuell Wechselgeld.

a) Ermitteln Sie eine möglichst kleine Zustandsmenge und erstellen Sie einen Zustandsgraphen für diesen Automaten.

b) Kodieren Sie die Zustände, Ein- und Ausgaben binär und ermitteln Sie in dieser Kodierung eine Schalttabelle (Wertetabelle) für die Ausgaben und neuen Zustände als Funktion von den Eingaben und alten Zuständen.

▹ Frage 2

Eine Schaltung mit zwei Eingängen und zwei Ausgängen enthalte drei Flip-Flops z_1, z_2 und z_3 und folgende Schaltfunktionen:

e_1	e_2	z_1^{alt}	z_2^{alt}	$z_3^{alt} a_1$	a_2	z_1^{neu}	z_2^{neu}	z_3^{neu}	
0	1	0	0	0	1	1	0	1	0
0	1	0	1	0	0	1	0	0	0
0	1	0	1	1	1	1	0	1	0
1	0	0	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	1
1	1	0	0	1	1	1	1	0	0
1	1	1	0	0	1	1	0	1	1

a) Stellen Sie die Schaltung durch einen Zustandsgraphen dar.

b) Stellen Sie die Schaltung durch möglichst effiziente boolesche Funktionen für die Bitwerte in den fünf Spalten rechts vom Doppelbalken dar.

▹ Frage 3

Im Druckertreiber wird der Befehl `drucke(druckauftrag);` ausgeführt. In einem anderen Programm werden die Befehle

```
vorbereiten();
druckauftrag=seite;
schreibe(‘‘Fertig!’’);
```

ausgeführt.

Die Variable `druckauftrag` ist im Druckertreiber und im Programm identisch. Zusätzlich seien die Semaphore `drucken` und `druckenfertig` vorhanden.

Erweitern Sie die Befehle für den Druckertreiber und das Programm um p- und v-Aufrufe[1], so daß der Drucker erst druckt, wenn ein Druckauftrag vorhanden ist, und das Programm erst “`Fertig!`” schreibt, wenn der Druck vollständig an den Drucker geliefert wurde.

[1] p-Aufrufe beanspruchen die übergebene Semaphore für sich oder warten, bis diese frei ist; v-Aufrufe geben die Semaphore wieder frei und rufen den nächsten wartenden Prozeß auf.

▹ Frage 4

Auf einem Computer laufen drei Computerprogramme, die Faxe erstellen und über einen ISDN-Anschluß verschicken. Der ISDN-Anschluß bietet zwei unabhängige gleichzeitig benutzbare Telefonleitungen.

Das eigentliche Versenden der Faxse wird von einem ***Spooler*** verrichtet, einem Programm, das Faxaufträge in einer Warteschlange sammelt und sie über eine Telefonleitung verschickt, falls eine frei ist. Die Warteschlange hat Platz für höchstens 20 Faxe.

a) Entwerfen Sie einen Monitor, um den Faxversand zu steuern. Das Einfügen von neuen Druckaufträgen in die Warteschlange und das Versenden der Faxse sind dabei Monitor-Funktionen.

   Hinweis: Nicht zu implementieren ist die Verwaltung, so daß eine Monitor-Methode zu einer Zeit nur einmal aufgerufen werden kann - gehen Sie davon aus, es gibt eine solche Monitor-Klasse, die dies bereits implementiert und die sie verwenden können. Für die Bedingungsvariablen gibt es ebenfalls bereits eine Klasse, die die Funktionalität implementiert – sie müssen hier nur die Methodenaufrufe `wait()`, um den Prozeß in die Warteschlange einzufügen, und `signal()`, um alle Prozesse aus der Warteschlange der Bedingungsvariable in die Hauptwarteschlange des Monitors zu überführen, in die entsprechenden Stellen in den Monitormethoden einfügen

b) Realisieren Sie diesen Monitor nur mit Hilfe von Semaphoren, und geben Sie an, wie die drei Faxe erstellende Programme und der Spooler die Semaphoren verwenden.

   Sie dürfen dabei Semaphoren einsetzen, die anstelle der »booleschen« Werte **ein** und **aus** natürliche Zahlen als Werte annehmen können, aber geben Sie an, wie die entsprechenden p- und v-Aufrufe mit diesen Semaphoren umgehen.

2. Lösungsvorschläge zu den Fragen

▹ Frage 1

Entwerfen Sie einen Fahrkartenautomaten für ein Nahverkehrsunternehmen. Es gibt zwei Fahrpreisstufen: Ticket A für 2 € oder Ticket B für 3 €. Der Automat hat eine Wahltaste für jede Fahrpreisstufe und zwei weitere Wahltasten, ob man für einen Aufpreis von 1 € die Karte für die Benutzung in der Stoßzeit vor 9 Uhr morgens validieren möchte.

Vor dem Einwurf des Geldes ist zuerst das gewünschte Ticket zu wählen, anschließend kann man eventuell die Gültigkeit vor 9 Uhr hinzubuchen. Danach kann dann das Geld eingeworfen werden. Die Maschine nimmt nur 1 €- und 2 €-Münzen an.

Hat der Kunde genügend Geld eingeworfen, erhält er die gewählte Fahrkarte und eventuell Wechselgeld.

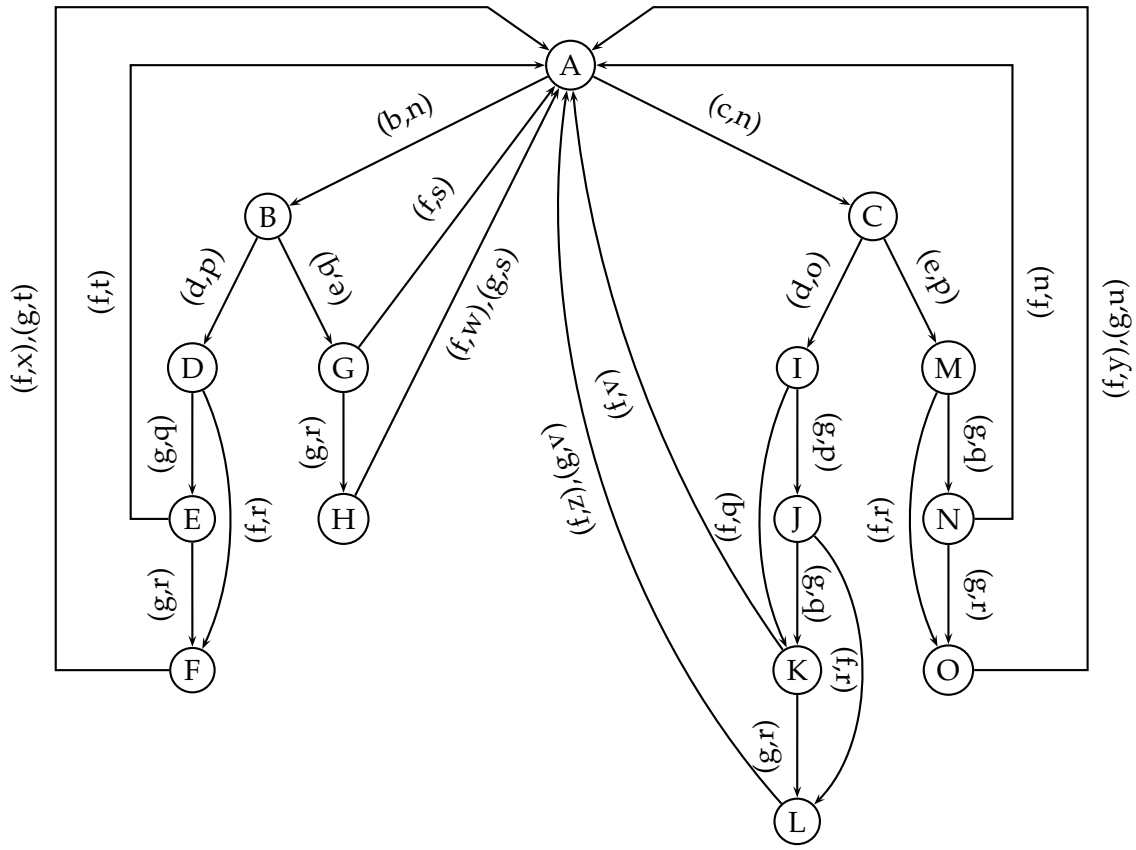
a) Ermitteln Sie eine möglichst kleine Zustandsmenge und erstellen Sie einen Zustandsgraphen für diesen Automaten.

b) Kodieren Sie die Zustände, Ein- und Ausgaben binär und ermitteln Sie in dieser Kodierung eine Schalttabelle (Wertetabelle) für die Ausgaben und neuen Zustände als Funktion von den Eingaben und alten Zuständen.

▹ Antwort 1

a)

Eingaben		Ausgaben		Zustände	
nichts	a	Ticket wählen (A/B)	m	Anfang	A
Ticket A	b	Stoßzeit-Ticket (j/n)	n	Ticket A	B
Ticket B	c	noch 4 €	o	Ticket B	C
Stoßzeit ja	d	noch 3 €	p	Ticket A Stoßzeit Schuld 3 €	D
Stoßzeit nein	e	noch 2 €	q	Ticket A Stoßzeit Schuld 2 €	E
2 €-Münze	f	noch 1 €	r	Ticket A Stoßzeit Schuld 1 €	F
1 €-Münze	g	Fahrkarte A	s	Ticket A Schuld 2 €	G
1 €-Münze	f	Fahrkarte A Stoßzeit	t	Ticket A Schuld 1 €	H
		Fahrkarte B	u	Ticket B Stoßzeit Schuld 4 €	I
		Fahrkarte B Stoßzeit	v	Ticket B Stoßzeit Schuld 3 €	J
		Fahrkarte A + 1 €	w	Ticket B Stoßzeit Schuld 2 €	K
		Fahrkarte A Stoßzeit + 1 €	x	Ticket B Stoßzeit Schuld 1 €	L
		Fahrkarte B + 1 €	y	Ticket B Schuld 3 €	M
		Fahrkarte B Stoßzeit + 1 €	z	Ticket B Schuld 2 €	N
				Ticket B Schuld 1 €	O



Zusätzlich sind noch die folgenden Null-Aktionen, d. h. Eingaben/Ausgaben, die zu keiner Zustandsänderung führen, möglich. Diese wurden wegen der Übersichtlichkeit nicht in den Zustandsgraphen mit eingezeichnet. Würde mandiese einzeichnen, hätte diese das Aussehen: (a,*alt*)

Null-Aktionen am Zustand			
	A→(a,m)	F→(a,r)	K→(a,q)
	B→(a,n)	G→(a,q)	L→(a,r)
	C→(a,n)	H→(a,r)	M→(a,p)
	D→(a,p)	I→(a,o)	N→(a,q)
	E→(a,q)	J→(a,p)	O→(a,r)

b)

Eingabe	Zustand alt	Ausgabe	Zustand neu
a	A	m	A
b	A	n	B
c	A	n	C
a	B	n	B
d	B	p	D
e	B	q	G
a	C	n	C
d	C	o	I
e	C	p	M
a	D	p	D
f	D	r	F
g	D	q	E
a	E	q	E
f	E	t	A
g	E	r	F
a	F	r	F
f	F	x	A
g	F	t	A
a	G	q	G
f	G	s	A
g	G	r	H
a	H	r	H
f	H	w	A
g	H	s	A
a	I	o	I
f	I	q	K
g	I	p	J
a	J	p	J
f	J	r	L
g	J	q	K
a	K	q	K
f	K	v	A
g	K	r	L
a	L	r	L
f	L	z	A
g	L	v	A
a	M	p	M
f	M	r	O
g	M	q	N
a	N	q	N
f	N	u	A
g	N	r	O
a	O	r	O
f	O	y	A
g	O	u	A

Eingabe	Ausgabe	Zustand	Bitwert
a	m	A	0000
b	n	B	0001
c	o	C	0010
d	p	D	0011
e	q	E	0100
f	r	F	0101
g	s	G	0110
	t	H	0111
	u	I	1000
	v	J	1001
	w	K	1010
	x	L	1011
	y	M	1100
	z	N	1101
		O	1110

e_1	e_2	e_3	z_1^{alt}	z_2^{alt}	z_3^{alt}	z_4^{alt}	a_1	a_2	a_3	a_4	z_1^{neu}	z_2^{neu}	z_3^{neu}	z_4^{neu}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
0	1	1	0	0	0	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	0	1	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0	0	1	0	1	0	0	0
1	0	0	0	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	1	1	0	0	1	1	0	0	1	1
1	0	1	0	0	1	1	0	1	0	1	0	1	0	1
1	1	0	0	0	1	1	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	0	1	0	1	0	0	0	1	1	1	0	0	0	0
1	1	0	0	1	0	0	0	1	0	1	0	1	0	1
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	0	1	1	0	0	0	0
1	1	0	0	1	0	1	0	1	1	1	0	0	0	0
0	0	0	0	1	1	0	0	1	0	0	0	1	1	0
1	0	1	0	1	1	0	0	1	1	0	0	0	0	0
1	1	0	0	1	1	0	0	1	0	1	0	1	1	1
0	0	0	0	1	1	1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	1	1	0	1	0	0	0	0	0
1	1	0	0	1	1	1	0	1	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	1	0	0	0
1	0	1	1	0	0	0	0	1	0	0	1	0	1	0
1	1	0	1	0	0	0	0	0	1	1	1	0	0	1
0	0	0	1	0	0	1	0	0	1	1	1	0	0	1
1	0	1	1	0	0	1	0	1	0	1	1	0	1	1
1	1	0	1	0	0	1	0	1	0	0	1	0	1	0
0	0	0	1	0	1	0	0	1	0	0	1	0	1	0
1	0	1	1	0	1	0	1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	0	1	0	1	1	0	1	1
0	0	0	1	0	1	1	0	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1	1	0	1	0	0	0	0
1	1	0	1	0	1	1	1	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	0	0
1	0	1	1	1	0	0	0	1	0	1	1	1	1	0
1	1	0	1	1	0	0	0	1	0	0	1	1	0	1
0	0	0	1	1	0	1	0	1	0	0	1	1	0	1
1	0	1	1	1	0	1	1	0	0	0	0	0	0	0
1	1	0	1	1	0	1	0	1	0	1	1	1	1	0
0	0	0	1	1	1	0	0	1	0	1	1	1	1	0
1	0	1	1	1	1	0	1	1	0	0	0	0	0	0
1	1	0	1	1	1	0	1	0	0	0	0	0	0	0

▹ Frage 2

Eine Schaltung mit zwei Eingängen und zwei Ausgängen enthalte drei Flip-Flops z_1, z_2 und z_3 und folgende Schaltfunktionen:

e_1	e_2	z_1^{alt}	z_2^{alt}	$z_3^{alt} a_1$	a_2	z_1^{neu}	z_2^{neu}	z_3^{neu}	
0	1	0	0	0	1	1	0	1	0
0	1	0	1	0	0	1	0	0	0
0	1	0	1	1	1	1	0	1	0
1	0	0	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	1
1	1	0	0	1	1	1	1	0	0
1	1	1	0	0	1	1	0	1	1

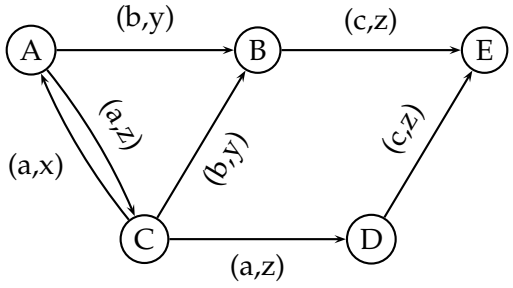
a) Stellen Sie die Schaltung durch einen Zustandsgraphen dar.

b) Stellen Sie die Schaltung durch möglichst effiziente boolesche Funktionen für die Bitwerte in den fünf Spalten rechts vom Doppelbalken dar.

▹ Antwort 2

a)

Eingang	Ausgang	Zustand	Bitfolge
-	-	A	000
a	x	B	001
b	y	C	010
c	z	D	011
-	-	E	100



b)

$$
\begin{aligned}
a_1 &= \overline{e_2 z_2^{alt} \overline{z_3^{alt}}} \\
a_2 &= e_2 \\
z_1^{neu} &= e_1 z_3^{alt} \\
z_2^{neu} &= z_2^{alt} z_3^{alt} + e_2 \overline{z_2^{alt}}\,\overline{z_3^{alt}} \\
z_3^{neu} &= \bar{e}_2 + z_1^{alt}
\end{aligned}
$$

▹ Frage 3

Im Druckertreiber wird der Befehl `drucke(druckauftrag);` ausgeführt. In einem anderen Programm werden die Befehle

```
vorbereiten();
druckauftrag=seite;
schreibe(‘‘Fertig!’’);
```

ausgeführt.

Die Variable `druckauftrag` ist im Druckertreiber und im Programm identisch. Zusätzlich seien die Semaphore `drucken` und `druckenfertig` vorhanden.

Erweitern Sie die Befehle für den Druckertreiber und das Programm um p- und v-Aufrufe[1], so daß der Drucker erst druckt, wenn ein Druckauftrag vorhanden ist, und das Programm erst “`Fertig!`” schreibt, wenn der Druck vollständig an den Drucker geliefert wurde.

▹ Antwort 3

Programm

```
vorbereiten();
p(drucken);
druckauftrag=seite;
v(drucken);
p(druckenfertig);
schreibe(Fertig!);
v(druckenfertig);
```

Druckertreiber

```
p(druckenfertig);  /* Initialisierung */
p(drucken);
if(druckauftrag)
{
   drucke(druckauftrag);
   v(druckenfertig);
   p(druckenfertig);
}
v(drucken);
```

Der Druckertreiber muß beim Starten die Semaphore `druckenfertig` besetzen, so daß das Programm nicht die Semaphore direkt erhalten kann und warten muß, bis der Druckertreiber wirklich fertig ist. Der Druckertreiber holt sich dann die Semaphore `drucken` und prüft anschließend, ob überhaupt Druckaufträge vorhanden sind (`if(druckauftrag)`); wenn ja, druckt er diesen, gibt die Semaphore `druckenfertig` frei, um sie direkt danach wieder zu beanspruchen. Anschließend wird, egal ob ein Druckauftrag vorhanden war oder nicht, das Drucken wieder freigegeben.

[1] p-Aufrufe beanspruchen die übergebene Semaphore für sich oder warten, bis diese frei ist; v-Aufrufe geben die Semaphore wieder frei und rufen den nächsten wartenden Prozeß auf.

▹ Frage 4

Auf einem Computer laufen drei Computerprogramme, die Faxe erstellen und über einen ISDN-Anschluß verschicken. Der ISDN-Anschluß bietet zwei unabhängige gleichzeitig benutzbare Telefonleitungen.

Das eigentliche Versenden der Faxse wird von einem ***Spooler*** verrichtet, einem Programm, das Faxaufträge in einer Warteschlange sammelt und sie über eine Telefonleitung verschickt, falls eine frei ist. Die Warteschlange hat Platz für höchstens 20 Faxe.

a) Entwerfen Sie einen Monitor, um den Faxversand zu steuern. Das Einfügen von neuen Druckaufträgen in die Warteschlange und das Versenden der Faxse sind dabei Monitor-Funktionen.

Hinweis: Nicht zu implementieren ist die Verwaltung, so daß eine Monitor-Methode zu einer Zeit nur einmal aufgerufen werden kann - gehen Sie davon aus, es gibt eine solche Monitor-Klasse, die dies bereits implementiert und die sie verwenden können. Für die Bedingungsvariablen gibt es ebenfalls bereits eine Klasse, die die Funktionalität implementiert – sie müssen hier nur die Methodenaufrufe `wait()`, um den Prozeß in die Warteschlange einzufügen, und `signal()`, um alle Prozesse aus der Warteschlange der Bedingungsvariable in die Hauptwarteschlange des Monitors zu überführen, in die entsprechenden Stellen in den Monitormethoden einfügen

b) Realisieren Sie diesen Monitor nur mit Hilfe von Semaphoren, und geben Sie an, wie die drei Faxe erstellende Programme und der Spooler die Semaphoren verwenden.

Sie dürfen dabei Semaphoren einsetzen, die anstelle der »booleschen« Werte **ein** und **aus** natürliche Zahlen als Werte annehmen können, aber geben Sie an, wie die entsprechenden p- und v-Aufrufe mit diesen Semaphoren umgehen.

▹ Antwort 4

a)
```
#define MAX_SPOOL_ENTRIES 20 /* Def. der Randbedingungen */
#define MAX_ISDN           2

class cMonitor : public VMonitor
{
   private:
      uint uiCntFreeSpoolEntries;
      uint uiCntFreeISDN;
      cSpooler spooler;
      cCondition nonfull;  /* Bedingung Warteliste nicht voll */
      cCondition nonempty; /* Bedingung Warteliste nicht leer */
      cCondition free;     /* Bedingung ISDN-Leitung frei */
   public:
      cMonitor() : uiCntFreeSpoolEntries(MAX_SPOOL_ENTRIES),
                   uiCntFreeISDN(MAX_ISDN)
      { /* weitere Initialisierungen */ }
      void put(void *ptr)
      {
         /* Warten auf freien Spoolereintrag */
         while (uiCntFreeSpoolEntries == 0)
         { nonfull.wait(); }
         spooler.put(ptr);
         --uiCntFreeSpoolEntries;
         nonempty.signal();
      }
```

```
        void get(void *ptr)
        {
            /* Warten auf Eintrag in Spooler */
            while (uiCntFreeSpoolEntries == MAX_SPOOL_ENTRIES)
            { nonempty.wait(); }
            /* Warten auf freie ISDN-Leitung */
            while (uiCntFreeISDN == 0)
            { nonfree.wait(); }
            ptr = spooler.get();
            ++uiCntFreeSpoolEntries;
            --uiCntFreeISDN;
            nonfull.signal();
        }
        void isdnready(void)
        {
            /* ISDN-Leitung freigeben */
            ++uiCntFreeISDN;
            nonfree.signal();
        }
};
```

C++-Klasse für die Aufgaben. Die Klasse `VMonitor` implementiert die Funktionalität des allg. Monitors (Hauptwarteschlange...), die Klasse `cSpooler` hält den Platz für die Faxaufträge (wo auch immer) und besitzt die Methoden `cSpooler::put(void *ptr)` und `cSpooler::get( void *ptr)` um einen Faxauftrag zu speichern oder zurückzugeben (in dem `ptr`). Die Klasse `cCondition` implementiert die Bedingungsvariable mit den Methoden `cCondition::wait()` und `cCondition::signal()`.

Die Initialisierung des Monitors wird über den Konstruktor `cMonitor::cMonitor()` durchgeführt. Programme können über die Methoden `cMonitor::put(void *ptr)` einen Faxauftrag hinzufügen. Der Treiber für die ISDN-Karte kann über `cMonitor::get(void *ptr)` sich einen Auftrag zum Versenden abholen; nachdem er fertig ist, ruft dieser die Methode `cMonitor::isdnready()` auf, um die ISDN-Leitung als frei zu makieren.

Die Randbedingungen (2 ISDN-Leitungen, 20 Warteschlangenplätze) sind hier als Präprozessordirektiven eingefügt.

b) Hierfür werden zwei Semaphoren benötigt:
   - *prog2spool* mit Wertebereich 0–20 für die Steuerung zwischen Programmen und Spooler;
   - *spool2isdn* mit Wertebereich 0–2 für die Steuerung zwischen Spooler und ISDN-Leitung.

   Programm ⇔ *Spooler*: Möchte ein Programm ein Fax versenden, so muß es sich eine Semaphore *prog2spool* besorgen. Dabei gibt diese Semaphore an, ob noch Platz in der Warteschlange ist. Ist dies der Fall, so dekrementiert diese den Zähler in der Semaphore. Ist der Zähler auf 0, so muß das Programm erst warten, bis wieder Platz in der Warteschlange vorhanden ist, d. h. bis die Semaphore einen Zählerwert größer 0 hat, nur dann kann die Semaphore zugeteilt werden.

   Spooler ⇔ *ISDN*: Der Spooler prüft, ob derzeit Einträge in der Warteschlange vorhanden sind. Ist dies der Fall, so muß er, um ein Fax zu versenden, sich die Semaphore *spool2isdn* besorgen (p-Aufruf). Bei jedem p-Aufruf wird der Zähler um 1 dekrementiert, wenn er nicht 0 ist; wenn er 0 ist, muß der Spooler warten bis eine Leitung frei ist, d. h. der Semapohren-Zähler größer 0 ist. Nachdem das Fax versendet worden ist, gibt der ISDN-Anschluß die Leitung wieder frei (v-Aufruf der Semaphore *spool2isdn*), wobei der Zähler der Semaphore inkrementiert wird. Anschließend löscht der Spooler den Faxauftrag aus

der Warteschlange und gibt die Semaphore *prog2spool* (v-Aufruf) frei, d. h. der Zähler dieser Semaphore wird um 1 inkrementiert. Damit können nun weitere Faxe in die Warteschlange eingefügt werden, wenn diese voll gewesen ist.

D. h. die Zähler in den beiden Semaphoren modellieren die noch verfügbaren Plätze (Warteschlange bzw. ISDN-Leitungen) und werden bei p-Aufrufen dekrementiert, wenn > 0 (ansonsten Warten) und bei v-Aufrufen inkrementiert.

Liste der Versionen

Version	Datum	Bearbeiter	Bemerkung
1.0	12.02.2007	Krä	Dokumenterstellung