Synchrone Botschaften

Prozesse kommunizieren und synchronisieren sich direkt miteinander,
bzw. über Kanäle, die höchstens eine Botschaft aufnehmen.

Operationen:

- **send (b):** blockiert bis Partnerprozess bereit ist, die Botschaft zu empfangen
- **receive (v)**: blockiert bis Partnerprozess bereit ist, eine Botschaft zu senden.

Wenn Sender und Empfänger bereit sind, wir die Botschaft übertragen; Zuweisung v := b;

send-receive-Paar ist **Datenübertragung und Synchronisationspunkt** zugleich

Ursprung: Communicating Sequential Processes (CSP) [C.A.R. Hoare, CACM 21, 8, 178]

Notation in CSP und Occam:

`p:    ...   q ! Ausdruck ...` sende den Wert des Ausdrucks an Prozess q

`q:    ...   p ? Variable ...` empfange von Prozess p einen Wert in der Variablen

auch mit verschiedenen Ports an einem Prozess und mit zusammengesetzten Botschaften:

```
p:    ...   q ! Port1 (a1,..,an) ...
q:    ...   p ? Port1 (v1,..,vn) ...
```

Selektives Warten

Verzweigung abhängig von Bedingungen und **Kommunikationsbereitschaft (Guard)**

```
if Bedingung1; p ! x-> Anweisung1
[] Bedingung2; q ? y-> Anweisung2
[] Bedingung3; r ? z-> Anweisung3
fi
```

Kommunikationsanweisung im Guard liefert
wahr, falls Partnerprozess bereit ist, **falsch**, falls er terminiert ist, **offen** sonst
Sind einige Guards wahr, wird einer gewählt, die Kommunikation und die Anweisung ausgeführt
Sind alle Guards falsch, wird die Selektion Ausführen einer Anweisung beendet.
Sonst wird gewartet, bis einer der beiden Fälle eintritt.

Selektive Schleifen werden wiederholt bis alle Guards falsch liefern:

```
do Bedingung1; p ! x-> Anweisung1
[] Bedingung2; r ? z-> Anweisung2
fi
```

Auch indizerte Selektion:

```
if (i: 1..n) Bedingung; p[i] ? v -> Anweisungen fi
```

Beispiel: Beschränkter Puffer:

```
do cnt < N; Prod ? buf[rear]  -> cnt++; rear = rear % N + 1;
[] cnt > 0; Cons ! buf[front] -> cnt--; front = front % N + 1;
od
```

Präfix-Summen mit synchronen Botschaften

Synchrone Kommunikation leistet **Übertragung des Datums und Synchronisation.**
Nur notwendige Synchronisation (vergl. Barrieren, PPJ-47)

```
const N := 6; var a [0:N-1] : int;

process Worker (i := 0 to N-1)                    ein Prozess für jedes Element
   var d := 1, sum, new: int

   sum := a[i];

                     {Invariante SUM: sum = a[i-d+1] + ... + a[i]}
   do d < N-1 ->

      if (i+d) < N -> Worker(i+d) ! sum fi            alter Wert nach rechts

      if (i-d) >= 0-> Worker(i-d) ? new; sum := sum + new fi
                                                      neuer Wert von links
      d := 2*d                                        Abstandsverdopplung
   od                                                 {SUM and d >= N-1}
end
```

Es gibt immer paarige Operationen **`Worker(i+d) ! sum`** und **`Worker(j-d) ? new`**
=> kein Deadlock

Client/Server-Schema mit synchronen Botschaften

Technik: für jede vom Server angebotene Operationsart Kommunikation über 2 Ports:

- oprReq für die Übertragung der Parameter
- oprRepl für die Übertragung der Antwort

Schema der Client-Prozesse:

```
process Client (I := 1 to N)
   ...
   Server ! oprReq (myArgs)
   Server ? oprRepl (myRes)
   ...
end
```

Schema des Server-Prozesses:

```
process Server ()
   ...
   do (c: 1..N) BedingungOpr1; Client[c] ? oprReq(oprArgs)
                  -> Anfrage bearbeiten ...
                     Client[c] ! oprRepl(oprResults)
   [] ebenso für weitere Operationen ...
   od
end
```

Verständnisfragen (1)

Grundbegriffe

1. Erklären Sie die Begriffe sequentielle parallele, verzahnte, nebenläufige Prozesse.
2. Beschreiben Sie die 2 Arten in Java Threads erzeugen (3 Schritte).
3. Geben Sie die wichtigsten Methoden der Klasse Thread an.
4. Erklären Sie die Begriffe Atomare Aktion, Art-most-once Eigenschaft
5. Wie wird Interferenz zwischen Prozessen definiert?

Monitore

6. Erklären Sie wie Monitore die 2 Synchronisationskonzepte realisieren.
7. Semantik von Bedingungsvariablen und Varianten dazu.
8. Aus welchen 3 Gründen können Prozesse am Monitor warten?
9. Beschränkter Puffer auch mit 1 Bedingungsvariablen?
10. Warum muss man bei signal-and-continue die Bedingung in Schleife prüfen?
11. Eigenschaften der Monitore in Java.
12. Unter welchen Umständen kann man notify statt notifyAll verwenden?
13. Wann gilt eine Monitor-Invariante?

Verständnisfragen (2)

14. Systematischer Monitor-Entwurf in 5 Schritten
15. Monitor-Invariante zum Leser/Schreiber-Schema
16. Strategien ausdrücken: Schreiber bevorzugen. Keinen bevorzugen.
17. Entwurfsmethode zur „Begegnung von Prozessen“
18. Wie werden Wartebedingungen und Weck-Operationen bei der Zählvariablenmethode eingefügt?

Barrieren

19. Abstandsverdopplung am Beispiel Präfixsummen
20. Barrieren-Regel
21. Baum-Barriere
22. Symmetrische, verteilte Barriere

Datenparallelität

23. Datenparallel Listenenden finden
24. Schleifen zu trapezförmigem Iterationsraum angeben
25. Abhängigkeitsvektoren, Zulässigkeit in sequentiellen Schleifen

Verständnisfragen (3)

26. SRP-Transformationen
27. Benutzung der Transformationsmatrizen
28. Mit welchen Transformationen kann man die innere Schleife parallelisieren, wenn die Abhängigkeitsvektoren (0,1) und (1,0) vorliegen?

Asynchrone Botschaften

29. Begriff des Kanals und seine Operationen
30. wichtige Kanal-Strukturen
31. Client/Server Kanal-Strukturen
32. Welches Problem tritt auf, wenn mehrere Prozesse jeweils von mehreren Kanälen empfangen?
33. Erklären Sie den Begriff Konversationsfolgen.
34. Welche Operationen führt ein Knoten bei Broadcast im Netz aus?
35. Welche Operationen führt ein Knoten bei Probe/Echo im Netz aus?
36. Wieviele Botschaften werden bei Probe/Echo versandt?

Verständnisfragen (4)

Botschaften in verteilten Systemen

37. Wie geht man in Java mit Ports und Sockets um?
38. Erläutern Sie das Worker-Paradigma.
39. Skizzieren Sie die Prozess-Schnittstellen für verteiltes Branch-and-Bound.
40. Erläutern Sie die Technik Terminierung im Ring.
41. Was sind die Aufgaben bei Methodenfernanrufen (RMI)?
42. Geben Sie die Entwicklungsschritte zur RMI-Implementierung an.

Synchrone Botschaften

43. Vergleichen Sie die Grundkonzepte synchroner und asynchroner Botschaften.
44. Erklären Sie das Konstrukt zu selektiven Warten bei synchronen Botschaften.
45. Warum sind Programme mit synchronen Botschaften weniger redundant als mit asynchronen Botschaften.
46. Skizzieren Sie einen Server zur Ressourcenvergabe nach dem Schema für synchrone Botschaften.