Input Statements

```
in op1 (...) and B1 by e1 -> S1;
[] ...
[] opn (...) and Bn by en -> Sn;
ni
```

- `Bi`: synchronization expression (optional)
- `ei`: scheduling expression (optional) – Aufrufe (der gleichen Operation) mit niedrigerem Wert haben Vorrang

Folie 1

Fragen:

(2) – was bedeutet es, wenn mehrere op_i gleich sind?

(3) – insbesondere mit scheduling expression?
wird ein op_i mit niedrigerem Wert
der gleiche Operation in anderer Zeile
mit höherem Wert vorgezogen?

(1) – welche Operation wird gewählt, wenn
mehrere möglich sind?

Auswertungsstrategie

```
in op1 (...) and B1 by e1 -> S1;
[] ...
[] opn (...) and Bn by en -> Sn;
ni
```

1. nimm ältesten Aufruf, für den eines der Statements
   `opi (...) and Bi` wahr ist
   (mehrere möglich: nimm erstes Statement)
2. gibt es *scheduling expression*:
   (a) berechne *scheduling value* von allen anderen Aufrufen von `opi`, für die `Bi` gültig ist
   (b) wähle Aufruf mit minimalem scheduling value (bei mehreren: den ältesten Aufruf mit minimalem Wert)

Beispiele

Angenommen, man hat den folgenden Stack von Aufrufen:

```
call b(2)  -- neuester Aufruf
call a(4)
call a(5)
call c(7)  -- aeltester Aufruf
```

Betrachte:

```
in b(m) by m -> ..
[] a(m) by m -> ..
[] a(m) by 0 -> ..
ni
```

2. $\rightsquigarrow$ `call a(4)`

Betrachte:

```
in b(m) -> ..
[] a(m) by 8-⌊m/2⌋ -> ..
ni
```

2. $\rightsquigarrow$ `call a(5)`

Folie 3

Linkes Programm:

- ältester Aufruf zuerst → c(7)
  kann nicht bearbeitet werden, nur a oder b
  daher wird der nächst-älteste geprüft
- a(5): kann bearbeitet werden, von 2. oder 3. Zeile
  nach Strategie von S.2: das erste "a(m) by m" wird
  gewählt
- da scheduling expression: Prüfung weiterer
  vorliegender a-Aufrufe
  a(4): hat niedrigeren scheduling Wert
  → wird vorgezogen

Resultat: 2. Zeile ("a(m) by m")
wird ausgeführt, Aufruf a(4)

Rechtes Programm

- ältester Aufruf zuerst: c(7) kann nicht
  behandelt werden
- a(5): kann bearbeitet werden
- Testen weiterer a-Aufrufe:
  a(4): hat gleichen scheduling Wert wie a(5)
  → folglich wird der ältere Aufruf gewählt,
  d.h. a(5)

Time Server 8.4'

Angenommen, man hat den folgenden Stack von Aufrufen:

```
delay (1)  -- neuester Aufruf
tick ()
tick ()
delay (2)  -- aeltester Aufruf
```

ohne scheduling expression by `waketime`:

1. `tick()`
2. `tick()`
3. `delay(2)`
4. `delay(1)`

mit scheduling expression by `waketime`: 3. und 4. vertauscht

Folie 4:

vorher Time server -Folie (Fig. 8.7)

Ohne scheduling expression (Fig. 8.7 original)

- initial tod = 0
- ältester Aufruf zuerst: delay(2)
  → kann nicht bearbeitet werden, da Boolesche Bedingung 2 <= tod nicht gilt
- nächst ältester Aufruf: das untere tick() wird bearbeitet, tod = 1
- delay(2) wieder nicht bearbeitbar
  → das 2. tick(), tod = 2
- delay(2) kann bearbeitet werden
- delay(1) wird bearbeitet

mit scheduling expression: by waketime

} genauso

- delay(2) kann bearbeitet werden, Prüfung anderer delay-Aufrufe
  → delay(1) hat Priorität und wird bearbeitet
- delay(2) wird bearbeitet

Beachte: trotzdem wird delay(1) erst bei tod = 2 ausgeführt !