

# Simple Programming Language

---

Markus Roggenbach

30. Oktober 2002

# Ein erstes Beispiel: MUX-SEM

**local  $y$ : integer where  $y = 1$**

$$P_1 :: \left[ \begin{array}{l} l_0 : \text{loop forever do} \\ \quad \left[ \begin{array}{l} l_1 : \text{noncritical} \\ l_2 : \text{request } y \\ l_3 : \text{critical} \\ l_4 : \text{release } y \end{array} \right] \end{array} \right] \parallel P_2 :: \left[ \begin{array}{l} m_0 : \text{loop forever do} \\ \quad \left[ \begin{array}{l} m_1 : \text{noncritical} \\ m_2 : \text{request } y \\ m_3 : \text{critical} \\ m_4 : \text{release } y \end{array} \right] \end{array} \right]$$

# Syntax

# Basic Statements

## skip

$$(u_1, \dots, u_n) := (e_1, \dots, e_n)$$

## await $c$

$c$  Bedingung;

Schritt nur wenn  $\llbracket c \rrbracket = \text{wahr}$

## halt $c$

$:=$  **await**  $F$

$\alpha \Leftarrow e$

sende Wert von  $e$  über Kanal  $\alpha$

$\alpha \Rightarrow u$

empfange Wert für  $u$  von Kanal  $\alpha$

**request**  $r$

$r := r - 1, r \geq 0$

**release**  $r$

$r := r + 1$

# Schematic Statements

**noncritical**

**critical**

**produce**  $x$   $x$  vom Typ integer

**consume**  $y$   $y$  vom Typ integer



**while**  $c$  **do**  $S$

**loop forever do**  $S$

**for**  $i := 1$  **to**  $m$  **do**  $S$

$:=$  **while**  $T$  **do**  $S$

$:=$  **i := 1;**

**while**  $i \leq m$

**do** [ $S; i := i + 1$ ]

$l : [l_1 : S_1 : \hat{l}_1 : ] || \dots || [l_k : S_k : \hat{l}_k : ]; \hat{l} :$

Kooperation

$[local\ declaration ; S]$

**local**  $y_1, \dots, y_k : T$  **where**  $\varphi$

Block

$\varphi$  hat die Form

$y_1 = e_1, \dots, y_n = e_n$

# Nicht unterbrechbare Anweisungen

*elementare sequentielle Anweisungen:*

Skip, Zuweisung, await, send, receive

*komplexe sequentielle Anweisungen:*

- when  $c$  do  $S$
- if  $c$  then  $S_1$  else  $S_2$
- $[S_1 \text{ or } \dots \text{ or } S_k]$
- $[S_1; \dots; S_k]$

*nicht unterbrechbare Anweisung:*  $\langle S \rangle$ ,  $S$  is sequentiell

# Kommunikation

nur in nicht unterbrechbaren Anweisungen!!!!

- synchron:  $\langle S_1; C; S_2 \rangle$ ,  
 $S_1$  und  $S_2$  ohne Kommunikation  
(dürfen auch fehlen)
- asynchron: jeder Kanal höchstens einmal in  $\langle \dots \rangle$

# Programme

$$P :: [\text{Deklarations-B; } [P_1 :: [l_1 : S_1; \hat{l}_1 :]] \mid \dots \mid [P_k :: [l_k : S_k; \hat{l}_k :]]]$$

*Deklarationsblock*: Folge von Deklarationen

*Deklaration*: Modus variable, . . . variable: Typ **where**  $\varphi_i$

*Modus: in, local, out*

*Zusicherungen  $\varphi_i$ :*

- optional
- **local, out:**  $y = e$  für initiale Werte  
Variablen in  $e$  sind vom Typ **in**
- **in:** beliebige Formel  
Einschränkung des Wertebereichs

*Vorbedingung eines Programms:  $\varphi_1 \dots \varphi_n$*

# Kanaldeklarationen

## *synchrone Kanäle*

Modus  $\alpha_1, \dots, \alpha_n$  : **channel of type**  
keine Speicherkapazität

## *asynchrone Kanäle*

Modus  $\alpha_1, \dots, \alpha_n$  : **channel [1..]** of type **where**  $\varphi$   
unbegrenzte Speicherkapazität

$\varphi$  : Anfangsbelegung

$\alpha_i = \langle \rangle$ , falls  $\varphi$  fehlt.