Concurrency: Theory, Languages and Programming

– CCS Example –

Session 5 – November 20, 2002

Uwe Nestmann

EPFL-LAMP

Value-Passing: Syntax

$\mathcal{N}$ channels

values

variables

$\mathcal{A}$ actions $\quad \overline{}$

"negative" actions $\overline{}$: *send name over channel .*

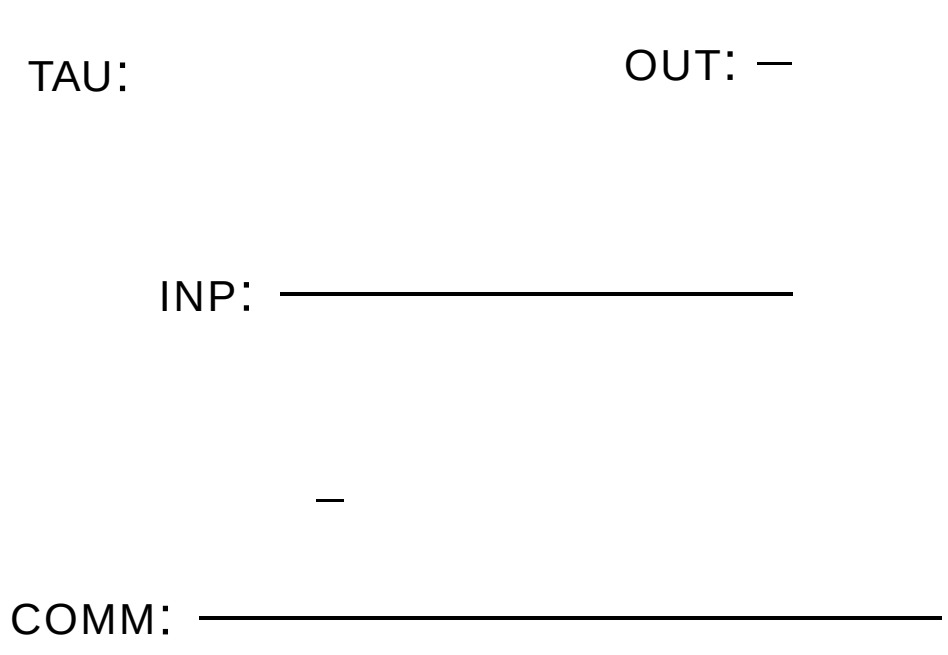
"positive" actions : *receive any value, say , over channel and "bind the result" to variable .*

Binding results in *substitution*
of the formal parameter by the actual parameter .

polyadic communication $\overline{}$ and (with pairwise different) *transmit many values at a time*.

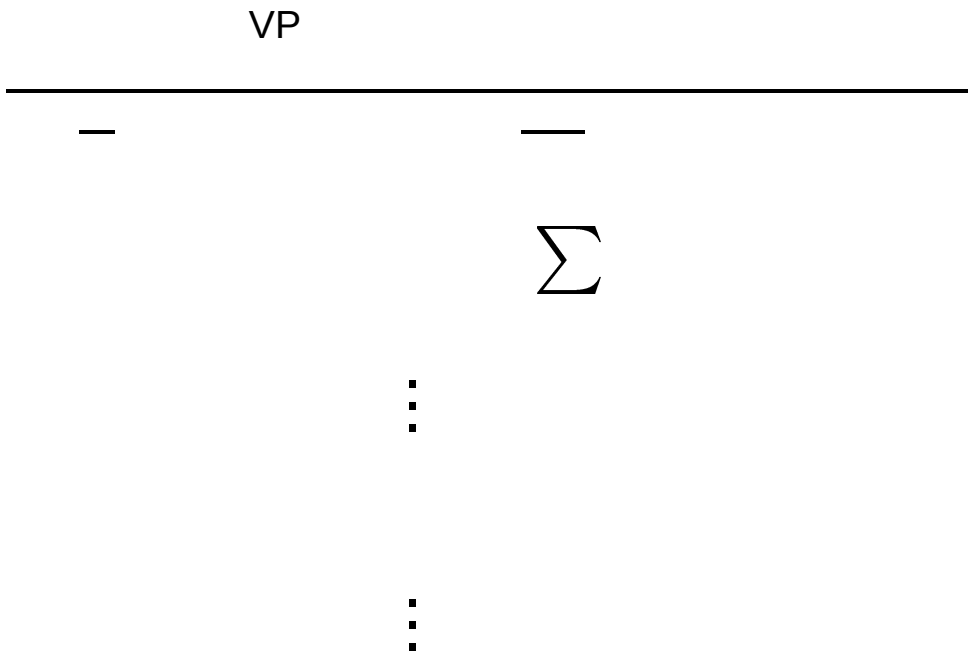
Value-Passing: Semantics I

directly: via LTS



Value-Passing: Semantics II

indirectly: via translation



Buffers in New Clothes . . .

$\mathcal{N}$ in out

Buff	in out 1-place buffer containing
Buff	in Buff
Buff	$\overline{\text{out}}$ Buff

Observe how much nicer name/value-passing is :-)

Bound and Free Names

and **bind** in

occurs **bound** in , if it occurs in a subterm or of

occurs **free** in , if it occurs without enclosing or in

Note the use of parentheses (round brackets).

Define and inductively on (sets of free/bound names of) . . .

Scheduler, Informally [Mil99, § 3.6]

a set of processes is to be scheduled

starts by signalling ¯ to the scheduler

completes by signalling ¯ to the scheduler

each *must not* run two tasks at a time

tasks of different *may* run at the same time

are required to occur cyclically (initially, starts)

for each , and must occur cyclically

maximal “progress”:

the scheduling must permit any of the buttons to be pressed at any time provided (1) and (2) are not violated.

Formal “Implementation” [§ 7.3]