DN2 / V2

Notiztitel 07.03.2005

Part I: Process Algebra.

Let Act be a denumerable set of "actions"

I.1. femto Process Algebra

A Syntax: Consider the language fPA (femto Process Algebra)

$$E ::= 0 \mid E + E \mid a.E \quad \text{with } a \in Act$$

(0: inaction, $E+E$: choice, $a.E$: prefix)

fPA is the set of all terms (expression) generated by this grammar.

Examples: $0,\ 0+0,\ (0+0)+0,\ a.0,\ (a.(b.(c.0)) + d.(b(a.0))$

Examples cont. $Act = \{get, put\}$

$Store = put.put.get.get.0$

$Store' = put.(get.0 + put.put.get.get.0)$

B] Semantics of fPA:

The semantics of a term $E \in fPA$ is the LTS $(fPA, Act, \longrightarrow, E)$ where $\longrightarrow \subseteq (fPA \times Act \times fPA)$ is the smallest relation satisfying the rules:

$$\langle \cdot \rangle \quad \frac{}{a.E \xrightarrow{a} E}$$

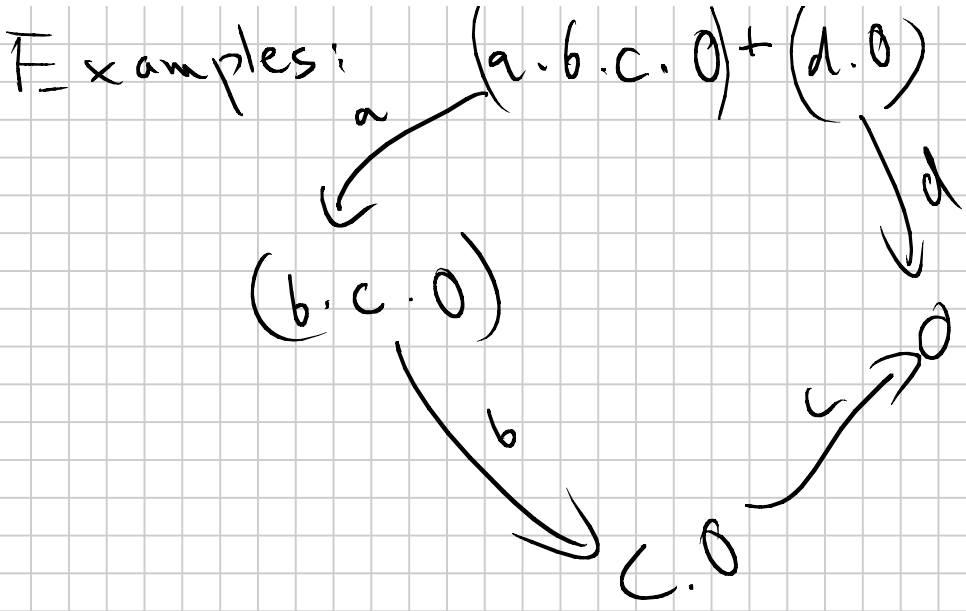
$$\langle +_R \rangle \quad \frac{E \xrightarrow{a} E'}{F+E \xrightarrow{a} E'}$$

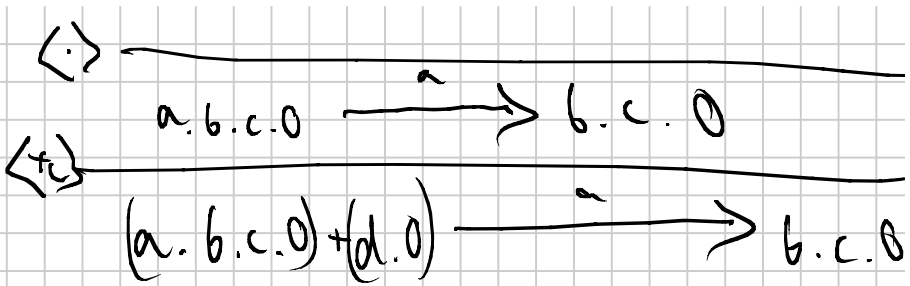
$$\langle +_L \rangle \quad \frac{E \xrightarrow{a} E'}{E+F \xrightarrow{a} E'}$$

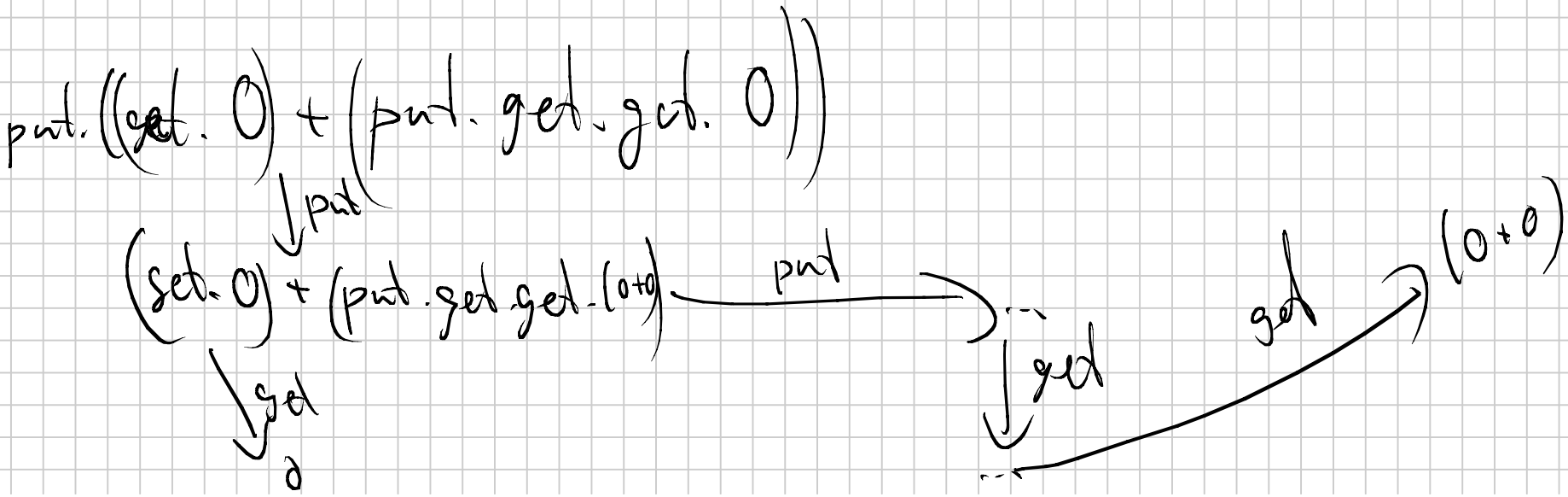
Structural operational semantics (Plotkin 81) operational behaviour is defined inductively over term structure

we write $E \xrightarrow{a} E'$ if $(E, a, E') \in \longrightarrow$

Examples:


(a.b.c.0)+(d.0)
a
d
(b.c.0)
b
c
c.0
0


⟨⟩
a.b.c.0
a
b.c.0
⟨+⟩
(a.b.c.0)+(d.0)
a
b.c.0


pwt.((get.0)+(pwt.get.get.0))
pwt
(set.0)+(pwt.get.get.(0+0))
pwt
get
0
get
(0+0)

I.2 pico Process Algebra

Let Var be a set of variables

A] Syntax: The language pPA is the set of all terms generated by

$$E ::= 0 \mid E + E \mid a.E \mid X \mid \text{rec } X : E$$

(X: variable; rec: recursion)

with $a \in Act$, $X \in Var$

Examples: $Var = \{X, Y, Z\}$

X, $\text{rec } X : a.X$, $\text{rec } X : \text{rec } Y : (Z + a.b.\text{rec } Y : c.Y)$

$\langle \cdot \rangle$ $\quad a.X \xrightarrow{a} X$ $\quad a \circlearrowright \text{rec } X : a.X$

$\langle \text{rec} \rangle$ $\quad \text{rec } X : a.X \xrightarrow{a} X\{\text{rec } X : a.X / X\}$

Expression vs. Processes.

Variables are means to introduce recursive (or repetitive) behaviour.

If in a term E a variable X occurs outside the scope of any binding "rec X", the variable is called a "free" variable in E.

A term, which contains free variables is called an "open" term.

Otherwise it is a "closed" term.

We like to restrict to the set of closed terms in a PA.

For pPA we call that subset $\subseteq$ pPA.

Closed terms are also called "processes".

Semantics:

The semantics of an expression $E \in pPA$ is the $LTS(pPA, Act, \rightarrow, E)$ where $\rightarrow \subseteq (pPA \times Act \times pPA)$ is the smallest relation satisfying

$$\frac{}{a.E \xrightarrow{a} E} \qquad \frac{E \xrightarrow{a} E'}{recX:E \xrightarrow{a} E'\{recX:E/X\}}\ [rec]$$

$$\frac{E \xrightarrow{a} E'}{E+F \xrightarrow{a} E'} \qquad \frac{E \xrightarrow{a} E'}{F+E \xrightarrow{a} E'}$$

$$0\{A/X\} := 0$$
$$X\{A/X\} := A$$
$$(a.E)\{A/X\} := a.(E)\{A/X\}$$
$$(E+F)\{A/X\} = E\{A/X\} + F\{A/X\}$$
$$(recY:E)\{A/X\} = recY: E\{A/X\}$$

$recX: rec Z: a.Z$

$recX: recY: a.Y$

$recX: recX: a.X$

- Operator precedence: "." binds strongest
  then "rec"
  then "+"
- Write "Stop" or "nil" for "0", or just omit it
- We use the terms "term" and "expression" interchangeably.
- We prefer to use mutually recursive equations as opposed to all these "rec's"

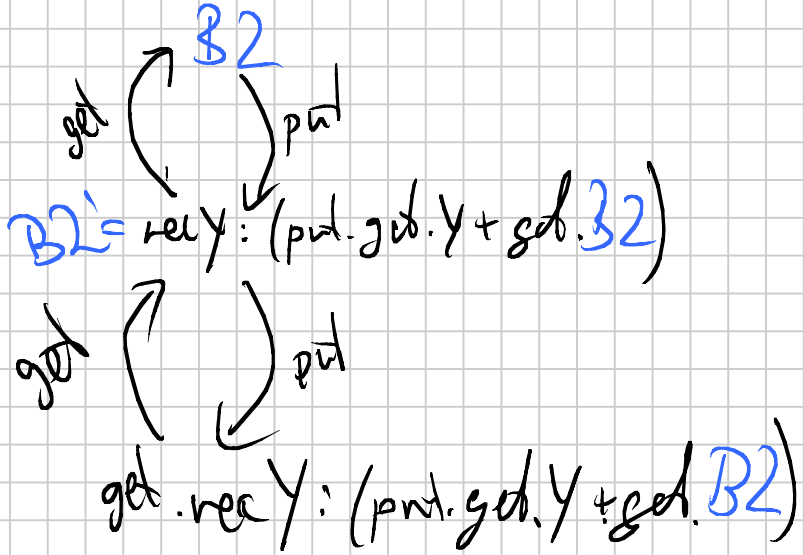
Example:
a.(b+c) means
a.(b.0 + c.0)

Examples: • A one-place buffer

$B1 = rec\ X : put.get.X$

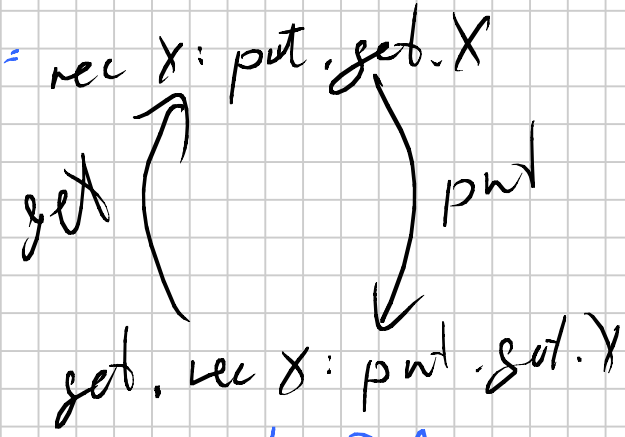
• A two-place buffer

$B2 = rec\ X : put.rec\ Y : (put.get.Y + get.X)$



$Act = \{put, get\}$

$B1 = rec\ X : put.get.X$



or: $get.B1$

$B1 = put.get.B1$

$B2 = put.B2'$

$B2' = put.get.B2' + get.B2$

I.3. nano Process Algebra

A] Syntax of nPA (nano Process Algebra)

$$0 \mid E+E \mid a.E \mid X \mid \text{rec } X : E \mid E \,|||\, E$$

asynchronous product

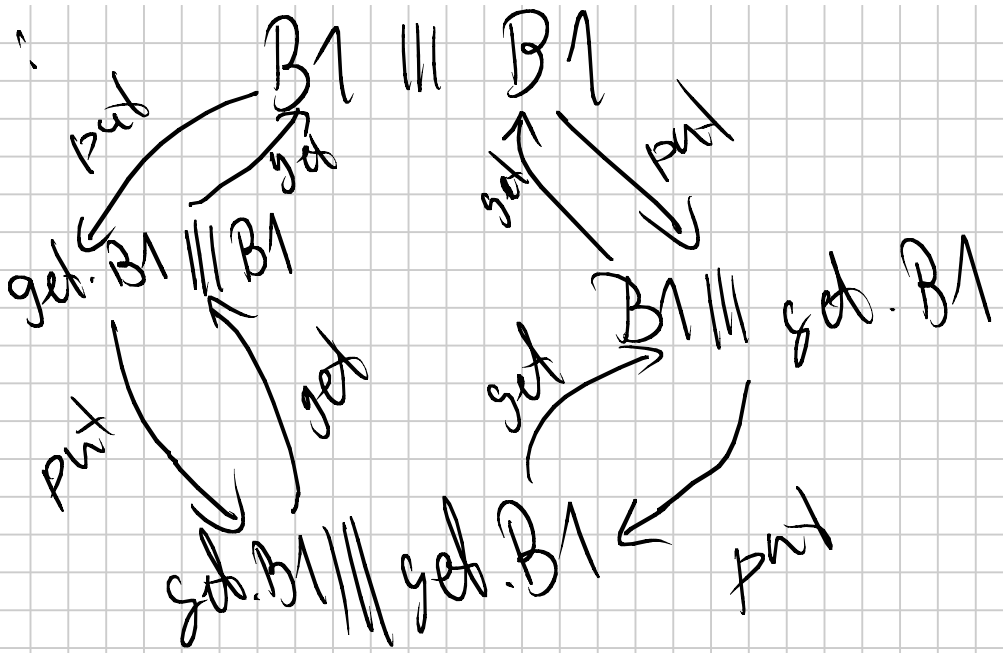
B] Semantics SOS as before, plus

$$\frac{E \xrightarrow{a} E'}{E \,|||\, F \xrightarrow{a} E' \,|||\, F}$$

$$\frac{E \xrightarrow{a} E'}{F \,|||\, E \xrightarrow{a} F \,|||\, E'}$$

This is the basic asynchronous process composition

Example:



$B1 = put.get.B1$

A] Syntax of synchronous nPA

$$0 \mid E+E \mid a.E \mid X \mid rec\,X:E \mid E \times E$$

B] Semantics SOS as for pPA, plus

$$\frac{E \xrightarrow{a} E' \;\wedge\; F \xrightarrow{b} F'}{E \times F \xrightarrow{s(a,b)} E' \times F'}$$

$s : Act \times Act \to Act$

Intermezzo: Synchronising two actions

Also in the asynchronous setting, we want to do certain things together (reading shared variables, receiving signals, RPC)

Principal options

$$s(a,b) = \begin{cases} \tau, & \text{if } \bar{a} = b \\ \bot, & \text{if } \bar{a} \neq b \end{cases}$$

requires pairs of "action/coaction" for all actions except "τ."

$$\bar{\bar{a}} = a$$

Successful Communication

CCS [Milner 89]

$$s(a,b) = \begin{cases} a, & \text{if } b = a \\ \bot, & \text{if } b \neq a \end{cases}$$

CSP [Hoare 78] - ACP

[ISO 8807, 87]

ACP: does not presuppose any specific function's

I.4. micro Process Algebra

Let $A \subseteq Act$

A] Syntax of μPA

$$0 \mid E+E \mid a.E \mid X \mid \text{rec } X : E \mid E|||E \mid E \|_A E$$

not essential

parallel composition

B] Semantics SOS as before, plus

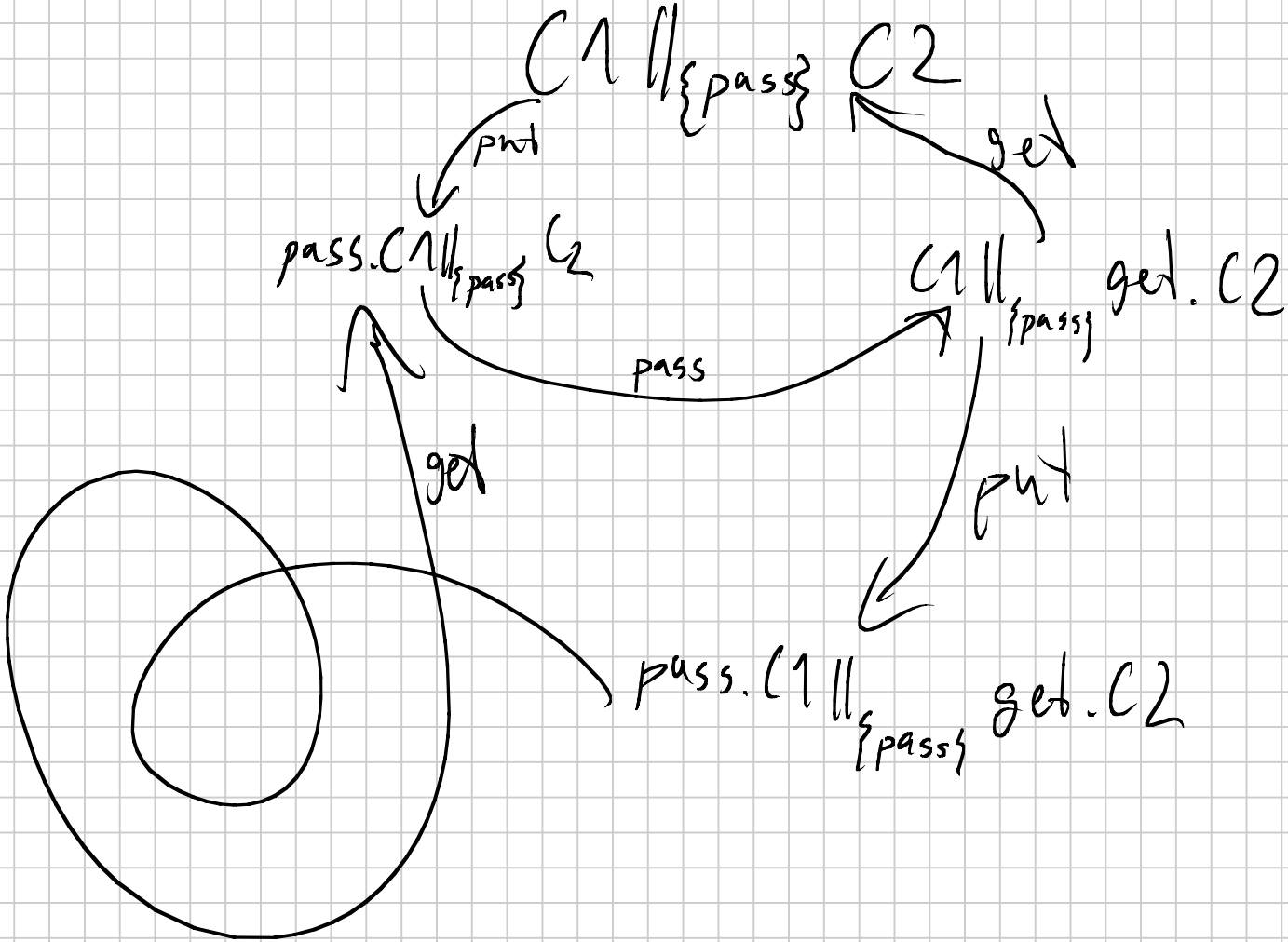
$$\frac{E \xrightarrow{a} E'}{E \|_A F \xrightarrow{a} E' \|_A F} \quad a \notin A$$

$$\frac{E \xrightarrow{a} E' \wedge F \xrightarrow{a} F'}{E \|_A F \xrightarrow{a} E' \|_A F'} \quad a \in A$$

$$\frac{E \xrightarrow{a} E'}{F \|_A E \xrightarrow{a} F \|_A E'} \quad a \notin A$$

Example: $recX : put.pass.X = C1$

$recY : pass.get.Y = C2$



Interme220: Synchronous vs. asynchronous communication

Here: Synchronous (Handshake) communication

Why not communication via channels (SPIN), or remote procedure call (RPC), or CORBA.

As noted by [Hoare] synchronous communication can encode the others, but not the converse.

I.5. milli Process Algebra

A] Syntax of mPA

$$0 \mid E+E \mid a.E \mid X \mid \mathrm{rec}X:E \mid E \|_A E \mid [E]_A \mid f(E)$$

[redacted] a distinguished action $\tau \in Act$ and $\tau \notin A$

B] Semantics SOS as before, plus:

$$\frac{E \xrightarrow{a} E'}{[E]A \xrightarrow{a} [E']A} \; a \in A$$

$$\frac{E \xrightarrow{a} E'}{[E]A \xrightarrow{a} [E']A} \; a \notin A$$

where $f: Act \to Act$ with $f(\tau) = \tau$

$$\frac{E \xrightarrow{a} E'}{f(E) \xrightarrow{f(a)} f(E')}$$

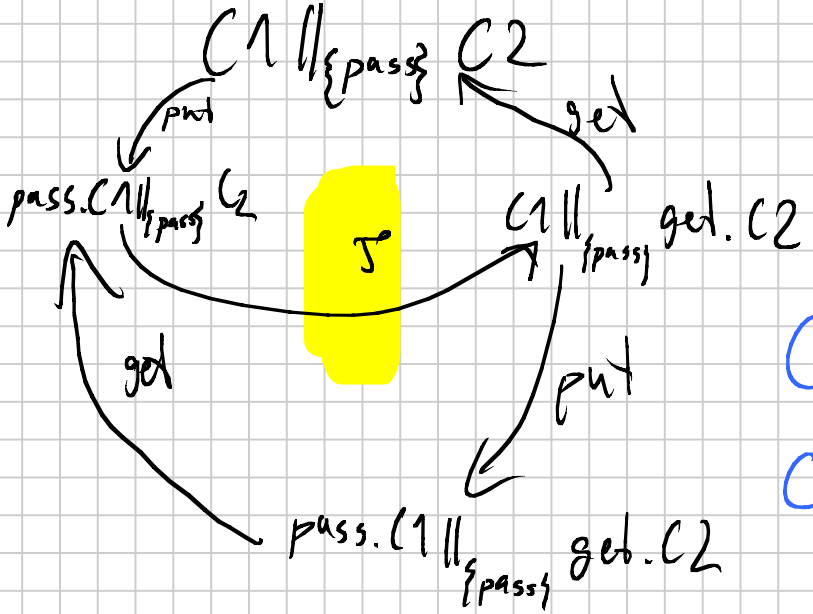
Example: $B1 = put.get.B1$

$$[get \mapsto pass](B1) \;\|_{\{pass\}}\; [put \mapsto pass](B1) \setminus \{pass\}$$

$C1$ $C2$

Notation

$$[put \mapsto pass](a) = \begin{cases} a, & \text{if } a \neq put \\ pass, & \text{sonst} \end{cases}$$



$C1 = put.pass.C1$

$C2 = pass.get.C2$