Typography in Process Algebra

Jos Baeten, Technische Universiteit Eindhoven, and
Jan Willem Klop, Vrije Universiteit, Amsterdam

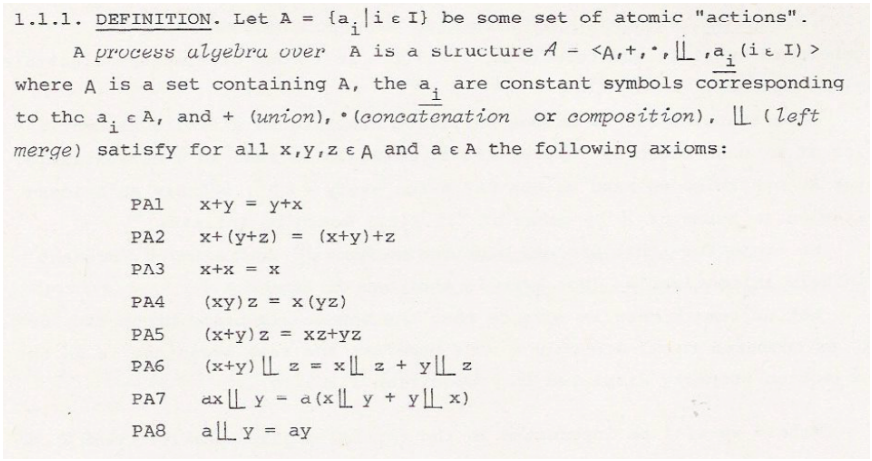
July 1, 2011

Abstract

This note is dedicated in gratitude to Jan Karel Lenstra, who reputedly is able to distinguish an ordinary full stop symbol (".") from its italicized version (".."), and who has inspired us with his knowledge and passion for typography. For his amusement we offer him a sample of typographical matters that we encountered in Process Algebra, of which the Amsterdam version originated at MC and CWI in the early eighties. The present note is written in the fonts Palatino and Euler, the latter being an upright cursive typeface, commissioned by the American Mathematical Society and designed and created by Hermann Zapf with the assistance of Donald Knuth.

1 History

The late seventies of the last century saw the advent of formal systems for communicating processes, an area then also designated as 'concurrency'. At CWI, after Jaco de Bakker introduced this new area in the Netherlands, Jan Bergstra and Jan Willem Klop endeavoured in the early eighties to develop a family of algebraic systems for various features of processes, specifically centering around ACP, Algebra of Communicating Processes. The phrase 'Process Algebra' was also coined then. Jan Willem wrote the first reports on process algebra on his trusted Olivetti with daisywheels, providing many symbols and fontsizes that could be produced on a typewriter. But not all symbols were available: the first report [17] introduced the 'left-merge' which in LaTeX can be presented as $\between$, and whose definition and employment marked our entrance in this field. Because of this auxiliary operator, process algebra is for a large part 'finitely axiomatizable', which is important for automation of the process algebra enterprise in systems such as mCRL. In [17], this symbol was written by hand, using India ink, see Figure 1.



1.1.1. DEFINITION. Let A = $\{a_i \mid i \in I\}$ be some set of atomic "actions".
A *process algebra over* A is a structure $\mathcal{A} = \langle A, +, \cdot, \between, \underline{a_i} (i \in I) \rangle$ where A is a set containing A, the $\underline{a_i}$ are constant symbols corresponding to the $a_i \in A$, and + (*union*), $\cdot$ (*concatenation* or *composition*), $\between$ (*left merge*) satisfy for all $x,y,z \in A$ and $a \in A$ the following axioms:

PA1	$x+y = y+x$
PA2	$x+(y+z) = (x+y)+z$
PA3	$x+x = x$
PA4	$(xy)z = x(yz)$
PA5	$(x+y)z = xz+yz$
PA6	$(x+y) \between z = x \between z + y \between z$
PA7	$ax \between y = a(x \between y + y \between x)$
PA8	$a \between y = ay$

Figure 1: *Olivetti and handwriting.*

For official purposes, all text written by CWI workers, either by typewriter or by hand, was converted by special typists into troff. They had to define a special symbol for the left-merge, which they called "Klop". See Figure 2, taken from [7].

Then, in 1985, Jan Bergstra and Jos Baeten moved to the University of Amsterdam, where there were no troff typists. Instead, they had Apple Macintosh computers and the typographical possibilities they offered. Some of us have fond memories of the MacPlus (Figure 3), the ImageWriter, the StyleWriter, MacWrite 5.0 and all that.

But, what to do with special process algebra symbols such as the left-merge? The group at University of Amsterdam set out to develop a special font for Process Algebra, using the Mac applications Fontastic and Fontographer. Two versions were developed: font Amsterdam, see Figure 4, based on Apple's font Times, and a sans-serif version called Brussel, based on Geneva. An example of a text in font Brussel is given in Figure 5.

The problem returned when laserwriters were introduced, then the special fonts could not be upgraded, but were still used in the books [1] and [13]. Later, the home-made process algebra typography was superseded by the superior LaTeX technology, and the rest of this note refers to that framework. One of us persisted until very recent years in using Mac typography with modern Mac applications such as Pages, but is now also converted to LaTeX.

$x+y = y+x$	A1	$a \lhd b = a$ if not $(a < b)$	P1
$x+(y+z) = (x+y)+z$	A2	$a \lhd b = \delta$ if $a < b$	P2
$x+x = x$	A3	$x \lhd yz = x \lhd y$	P3
$(x+y)z = xz+yz$	A4	$x \lhd (y+z) = (x \lhd y) \lhd z$	P4
$(xy)z = x(yz)$	A5	$xy \lhd z = (x \lhd z)y$	P5
$x+\delta = x$	A6	$(x+y) \lhd z = x \lhd z + y \lhd z$	P6
$\delta x = \delta$	A7		
$a \mid b = b \mid a$	C1		
$(a \mid b) \mid c = a \mid (b \mid c)$.	C2		
$\delta \mid a = \delta$	C3		
$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y$	CM1	$\theta(a) = a$	TH1
$a \lfloor\!\lfloor x = ax$	CM2	$\theta(xy) = \theta(x)\theta(y)$	TH2
$ax \lfloor\!\lfloor y = a(x \parallel y)$	CM3	$\theta(x+y) = \theta(x) \lhd y + \theta(y) \lhd x$	TH3
$(x+y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	CM4		
$(ax) \mid b = (a \mid b)x$	CM5		
$a \mid (bx) = (a \mid b)x$	CM6		
$(ax) \mid (by) = (a \mid b)(x \parallel y)$	CM7		
$(x+y) \mid z = x \mid z + y \mid z$	CM8		
$x \mid (y+z) = x \mid y + x \mid z$	CM9		
$\partial_H(a) = a$ if $a \notin H$	D1		
$\partial_H(a) = \delta$ if $a \in H$	D2		
$\partial_H(x+y) = \partial_H(x) + \partial_H(y)$	D3		
$\partial_H(xy) = \partial_H(x)\partial_H(y)$	D4		

Figure 2: *ACP, Algebra of Communicating Processes, with priorities.*

Figure 3: *1988 MacintoshPlus ED, still alive and well.*

Font name: Amsterdam
Font size: 12
Font number: 150
Location: System :System

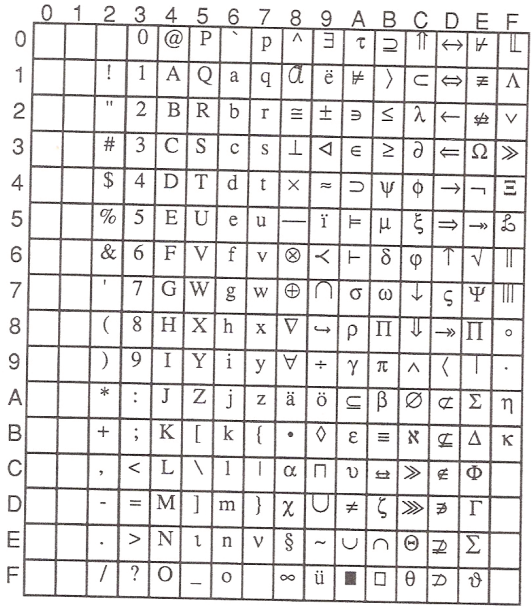
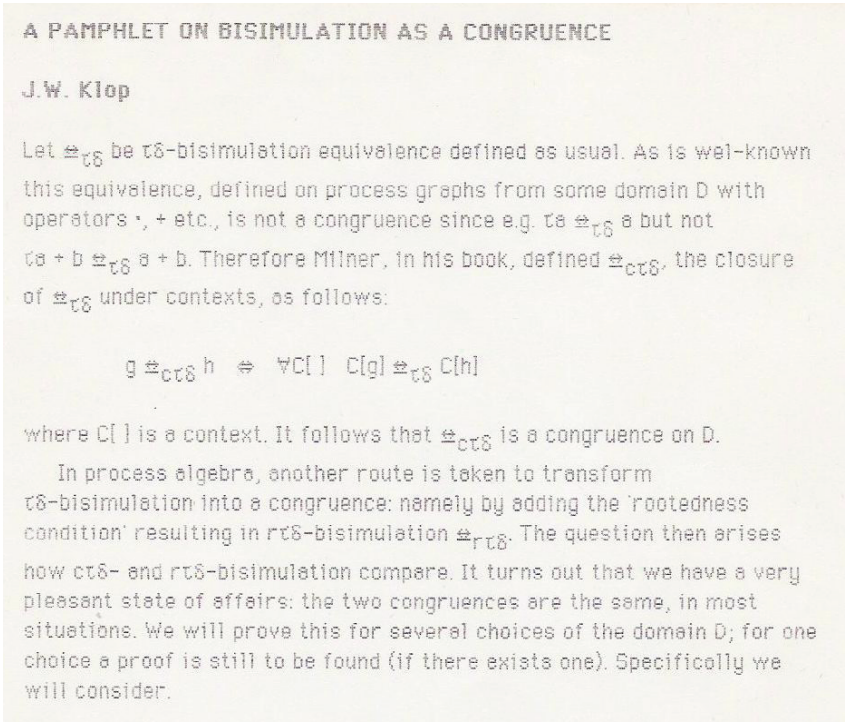


Figure 4: *process algebra font Amsterdam*.



A PAMPHLET ON BISIMULATION AS A CONGRUENCE

J.W. Klop

Let $\leftrightarrows_{\tau\delta}$ be $\tau\delta$-bisimulation equivalence defined as usual. As is well-known this equivalence, defined on process graphs from some domain D with operators $\cdot$, + etc., is not a congruence since e.g. $\tau a \leftrightarrows_{\tau\delta} a$ but not $\tau a + b \leftrightarrows_{\tau\delta} a + b$. Therefore Milner, in his book, defined $\leftrightarrows_{c\tau\delta}$, the closure of $\leftrightarrows_{\tau\delta}$ under contexts, as follows:

$$g \leftrightarrows_{c\tau\delta} h \quad \Leftrightarrow \quad \forall C[\,]\ \ C[g] \leftrightarrows_{\tau\delta} C[h]$$

where C[ ] is a context. It follows that $\leftrightarrows_{c\tau\delta}$ is a congruence on D.

In process algebra, another route is taken to transform $\tau\delta$-bisimulation into a congruence: namely by adding the 'rootedness condition' resulting in $r\tau\delta$-bisimulation $\leftrightarrows_{r\tau\delta}$. The question then arises how $c\tau\delta$- and $r\tau\delta$-bisimulation compare. It turns out that we have a very pleasant state of affairs: the two congruences are the same, in most situations. We will prove this for several choices of the domain D; for one choice a proof is still to be found (if there exists one). Specifically we will consider.

Figure 5: *A 'Pamphlet' from 1986 discussed in the PAM seminar (Process Algebra Meeting), which was held from 1985 until 2010.*

Figure 6: *Interrobang in Palatino Linotype, taken from Wikipedia.*

2 Punctuation symbols

Punctuation symbols are used in process algebra extensively. Well-known is the use of the period to denote action prefix, so e.g. $a.x$ means that action a is prefixed to term x, so first a is executed, followed by x, see CCS [23]. For sequential composition, the semicolon is sometimes used, so $x;y$ means x is followed by y, see CSP [21] or χ [15].

The exclamation mark is used for replication in the π-calculus [25], so $!x = x \between !x$, often called *bang*.

Besides this, it stands for a *send*, in CSP [21], but also in [2] (or is it [3]?), so action $c!d$ means that data element d is sent at communication port c.

A datum sent can also be received, and the question mark is used for this, so $c?d$ means that data element d is received at communication port c. In ACP-style process algebra, the simultaneous execution of matching communicating actions in parallel components is the resulting communication, so the simultaneous communication of $c!d$ and $c?d$ becomes the communication $c⁉d$, see [2] or [3].

The symbol ⁉, see Figure 6, is the *interrobang* (interrogation mark plus bang) invented by Martin K. Speckter in 1962 (see [29])).

It is used to ask a question in an excited manner, expresses excitement or disbelief in the form of a question, or asks a rhetorical question. It was popular in the 1960s, but its use declined soon after. Nowadays, we would write !? or ?! instead.

The asterisk stands for iteration or Kleene star, studied in process algebra in [18] or [10]. In x^*, either immediate termination takes place, or x is executed once, again followed by x^*. The vertical bar is used in CCS for parallel composition [23], while most other process algebras use the double vertical bar $\between$ for this. In ACP-style process algebra, the single bar is used for communication merge: the parallel composition of terms that starts with a communication action. The backslash $\backslash$ is used in CCS for restriction (also called encapsulation) [23], sometimes the 'forward' slash / is used for abstraction.

3 Greek symbols

Many Greek letters are used in process algebra. The letter α sometimes stands for the *alphabet* of a process, see [9]. Often, α is also used as ranging over a given set of actions plus some other actions. In [9], the letter β is used for an approximation of an alphabet, that is easier to calculate. The letter γ stands for the communication function, indicating which actions match in order to form a communication action together, probably because the letter occupies the same place in the Greek alphabet as the letter c in the Latin alphabet.

The letter δ is a constant standing for inaction, unsuccessful termination or deadlock in ACP-style, taking the name from the first letter of the word deadlock [19]. Since it is the neutral element of alternative composition, it was later renamed to 0 in [2]. The letter δ is also used for delay, see e.g. [24]. The capital letter Δ stands for divergence, see [20]. The letter ε is the constant standing for the empty process, successful termination or skip [30]. Since it is the neutral element of sequential composition, it was renamed to 1 in [2].

A use of the letter ζ in process algebra could not be found. The letter η was introduced for the silent step in [12], probably for no other reason than that this was an unused letter up to that time. The priority operator introduced in [8] uses the letter θ, also for no other reason than that. When in [2] a new auxiliary operator was used in the axiomatization of the priority operator, ϑ was used. The letter ι stands for an *idle* transition, e.g. in [22]. The letter κ is often used in ordinal arithmetic, but no use in process algebra was found.

The letter λ is used for the *state operator* and stems from [4]. Also a generalized version comes from here, and uses the letter Λ. The letter μ is used for a fixed point of recursion in CCS, already in [23]. The letter ν stands for now and is used for a process starting with no time delay in [5], but also stands for new and is used for a fresh variable in π-calculus [25]. In [4], the ν was used for a *localization* operator.

We have seen no uses of ξ or Ξ so far, however see the item χ further on. The omikron doesn't distinguish itself from the Latin character, so does not appear as itself. Lower case π of course names the π-calculus [25], and is used for projection in ACP-style process algebra [19]. Upper case Π is sometimes used for generalized parallel composition. The letter ρ is used for renamings in [4].

The letter σ is used for a time step, see e.g. [6] or for a time delay [11]. The capital Σ is widely used for generalized alternative composition or

sum, since [23]. The letter τ is universally used for a silent or unobservable step, dating back to [23]. As there, τ is put directly for the result of two matching communicating actions, it is the trace of a communication, and this is the origin of the name. In timed process algebra, the υ is used for time-out and time initialization, see [11].

The ϕ-calculus is a hybrid process algebra developed in [27]. The letter χ is also the name of a hybrid process algebra [15], taking its name from the initial of the word hybrid. At the moment, the language χ is being updated and renewed, with as working title Xí [14]. Reading this as Greek is the name of the capital χ, but reading it as Latin, Xí, it becomes Ξ. The X refers to the neXt operator of temporal logic. The ψ-calculus is an extension of the π-calculus with data and assertions about data, see [16]. Finally, ω is a special action denoting success in a testing semantics [26], but also is a calculus for mobile ad-hoc networks [28].

4 Other symbols

We conclude with some symbols not yet mentioned above.

left-merge We already discussed the left-merge $\between$, where $x \between y$ is the interleaving of processes x and y with the proviso that the first step has to be done by the left process x, hence the name left-merge. Typographically, the modern version of this symbol differs a bit from the ancient one, where the horizontal stroke was longer, see Figure 2. Many versions of the left-merge exist in LaTeX, such as $\between$, used in [2, 3].

encapsulation An important operator when composing processes is the *encapsulation operator* written as ∂. The reason for choosing this symbol was its association in some areas of mathematics as a boundary operator. It is parametrized by a set of actions H, so that $\partial_H(x)$ means that process x is not allowed to communicate with the environment using actions in the set H.

bisimulation Finally, there is the symbol for *bisimilarity* $\leftrightarrows$. It occurs not in the syntax of process algebra, but in its semantics, connecting the formal syntax, strings of symbols, with the real processes, mathematical entities. Processes x and y are bisimilar, $x \leftrightarrows y$, when roughly said, they can simulate each other.

References

[1] J.C.M. Baeten. *Procesalgebra*. Programmatuurkunde. Kluwer, 1986.

[2] J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra (Equational Theories of Communicating Processes)*. Number 50 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2009. USA Edition.

[3] J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra (Equational Theories of Communicating Processes)*. Number 50 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2010. UK Edition.

[4] J.C.M. Baeten and J.A. Bergstra. Global renaming operators in concrete process algebra. *Information and Computation*, 60(1/3):205–245, 1988.

[5] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[6] J.C.M. Baeten and J.A. Bergstra. Discrete Time Process Algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.

[7] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Proces algebra met interrupt mechanisme. In *Proceedings NGI-SION Informatica Symposium*, pages 129–135, 1985.

[8] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and Defining Equations for an Interrupt Mechanism in Process Algebra. *Fundamenta Informaticae*, IX(2):127–168, 1986.

[9] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Conditional Axioms and α/β-Calculus in Process Algebra. In M. Wirsing, editor, *Formal Description of Programming Concepts - III, IFIP Conference, Proceedings*, pages 77–103. Elsevier, 1987.

[10] J.C.M. Baeten, F. Corradini, and C.A. Grabmayer. A characterization of regular expressions under bisimulation. *Journal of the ACM*, 54(2):6.1–28, 2007.

[11] J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 2002.

[12] J.C.M. Baeten and R.J. van Glabbeek. Another look at abstraction in process algebra. In Th. Ottman, editor, *Automata, Languages and Programming, 14th Colloquium, ICALP 1987, Proceedings*, number 267 in Lecture Notes in Computer Science, pages 84–94. Springer-Verlag, 1987.

[13] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[14] Jos C.M. Baeten and Michel A. Reniers. Xí — a hybrid process algebra for cyber-physical systems. Draft., 2011.

[15] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Syntax and consistent equation semantics of hybrid chi. *Journal of Logic and Algebraic Programming*, 68(1-2):129–210, 2006.

[16] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *Proceedings LICS 2009*, pages 39–48, 2009.

[17] J. A. Bergstra and J. W. Klop. Fixed Point Semantics in Process Algebra. Technical Report IW 206, Mathematical Centre, Amsterdam, the Netherlands, 1982.

[18] J.A. Bergstra, W.J. Fokkink, and A. Ponse. Process Algebra with Recursive Operations. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 333–389. Elsevier, 2001.

[19] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information and Control*, 78(3):109–137, 1984.

[20] J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Failures without Chaos: a new Process Semantics for Fair Abstraction. In M. Wirsing, editor, *Formal Description of Programming Concepts - III, IFIP Conference, Proceedings*, pages 77–103. North-Holland, 1987.

[21] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[22] A.S. Klusener. *Models and Axioms for a Fragment of Real Time Process Algebra*. PhD thesis, Eindhoven University of Technology, Department of Computer Science, 1993.

[23] R. Milner. *A Calculus of Communicating Systems.* Number 92 in Lecture Notes in Computer Science. Springer Verlag, 1980.

[24] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[25] R. Milner. *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.

[26] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[27] William C. Rounds and Hosung Song. The phi-calculus: A language for distributed control of reconfigurable embedded systems. In *HSCC'03*, pages 435–449, 2003.

[28] Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. In *Proceedings of the 10th international conference on Coordination models and languages*, COORDINATION'08, pages 296–314. Springer-Verlag, 2008.

[29] Martin K. Speckter. *Disquisition on the Composing Stick*. Typophiles, Inc., 1971.

[30] J.L.M. Vrancken. The Algebra of Communicating Processes with Empty Process. *Theoretical Computer Science*, 177(2):287–328, 1997.