Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Bergstra, J.W. Klop

Algebra of communicating processes with abstraction

Department of Computer Science Report CS-R8403 January

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

ALGEBRA OF COMMUNICATING PROCESSES WITH ABSTRACTION

J.A. BERGSTRA, J.W. KLOP
Centre for Mathematics and Computer Science, Amsterdam

We present an axiom system ACP_τ for communicating processes with silent actions ('τ-steps'). The system is an extension of ACP, Algebra of Communicating Processes, with Milner's τ-laws and an explicit abstraction operator. By means of a model of finite acyclic process graphs for ACP_τ syntactic properties such as consistency and conservativity over ACP are proved. Furthermore the Expansion Theorem for ACP is shown to carry over to ACP_τ. Finally, termination of rewriting terms according to the ACP_τ axioms is proved using the method of recursive path orderings.

69F11, 69F12, 69F32, 69F43

1980 MATHEMATICS SUBJECT CLASSIFICATION: 68B10, 68C01, 68D25, 68F20.
1982 CR. CATEGORIES: F.1.1, F.1.2, F.3.2, F.4.3.
KEY WORDS & PHRASES: concurrency, communicating processes, internal actions, process algebra, bisimulation, process graphs, handshaking, terminating rewrite rules, recursive path ordering.
NOTE: This report will be submitted for publication elsewhere.

Report CS-R8403
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

INTRODUCTION

The equational theory ACP_τ is an integration of ACP (Algebra of Communicating Processes) and Milner's τ-laws. This paper studies the finite processes according to ACP_τ, i.e. the initial model of ACP_τ. In particular the following aspects are considered:

(i) Construction of a model of finite acyclic process graphs (modulo bisimulation) for ACP_τ.

(ii) A proof that the model of (i) is in fact the initial model of ACP_τ; stated in different terms this amounts to the soundness and completeness of ACP_τ for finite processes.

(iii) Analysis of a reduction system related to ACP_τ: using recursive path orderings termination of the reduction system is shown.

(iv) A proof of the Expansion Theorem.

(v) A proof of the associativity of parallel composition.

The paper is virtually self-contained, though some proofs make use of propositions shown in [3].

Related literature. ACP_τ was defined in [4]; the subsystem ACP was defined in [2]. Abstraction was studied in [3]. The formulation of the Expansion Theorem is taken from [5].

Both ACP and ACP_τ have been derived from Milner's CCS ([12]). In particular CCS contains the operators $+, ||, a\bullet$ for each atom a and derives as laws: A1,A2,A3 and T1,T2,T3. The axioms C1,C2 are from HENNESSY [10]; WINSKEL [13] surveys communication formats of atomic actions. The operator $\bullet$ is present in Hoare's CSP [11] as ';' and in DE BAKKER & ZUCKER [1] as '$\circ$'. We refer to GRAF & SIFAKIS [9] for a proof-theoretic discussion of the τ-laws. BROOKES & ROUNDS [6] contains an explicit description of bisimulation modulo τ on finite graphs.

The structure of this paper is as follows:

1. THE AXIOM SYSTEM ACP_τ

Let A be a finite set of atomic actions, containing a constant δ, and let $.|. : A \times A \longrightarrow A$ be a communication function which is commutative and associative and for which $\delta|a = \delta$. A communication $a|b = c$ is said to be proper if $c \neq \delta$. Further we consider the constant τ, for the silent action; we write $A_\tau = A \cup \{\tau\}$. Silent actions are obtained from applications of the abstraction operator τ_I which renames atoms $\in I \subseteq A$ into τ.

The signature of the equational theory ACP_τ is as follows:

$+$	*alternative composition (sum)*
$\bullet$	*sequential composition (product)*
$\|\|$	*parallel composition (merge)*
$⌊⌊$	*left-merge*
$\|$	*communication merge*
∂_H	*encapsulation*
τ_I	*abstraction*
δ	*deadlock / failure*
τ	*silent action*

Table 1.

Here the first five operators are binary, ∂_H and τ_I are unary. The operation ∂_H renames the atoms in H into δ and τ_I renames the atoms in I into τ.Here H and I are subsets of A_τ; in fact $H \subseteq A$ and $I \subseteq A-\{\delta\}$ (since we do not want to rename τ into δ or conversely).

The communication function | is extended to the communication merge, having the same notation, between processes (i.e. elements of a model of ACP_τ).

The left column in Table 2 (next page) is the axiom system ACP (without τ). In Table 2, 'a' varies over A.

The axioms T1,2,3 are the 'τ-laws' from MILNER [12].

Notation: often we will write xy instead of $x \bullet y$.

The initial algebra of the equational theory ACP_τ in Table 2 is called A_τ^ω.

ACP_τ

$x + y = y + x$	A1	$x\tau = x$	T1
$x + (y + z) = (x + y) + z$	A2	$\tau x + x = \tau x$	T2
$x + x = x$	A3	$a(\tau x + y) = a(\tau x + y) + ax$	T3
$(x + y)z = xz + yz$	A4		
$(xy)z = x(yz)$	A5		
$x + \delta = x$	A6		
$\delta x = \delta$	A7		
$a\|b = b\|a$	C1		
$(a\|b)\|c = a\|(b\|c)$	C2		
$\delta\|a = \delta$	C3		
$x\|\|y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x\|y$	CM1		
$a \lfloor\!\lfloor x = ax$	CM2	$\tau \lfloor\!\lfloor x = \tau x$	TM1
$(ax) \lfloor\!\lfloor y = a(x\|\|y)$	CM3	$(\tau x) \lfloor\!\lfloor y = \tau(x\|\|y)$	TM2
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	CM4	$\tau\|x = \delta$	TC1
$(ax)\|b = (a\|b)x$	CM5	$x\|\tau = \delta$	TC2
$a\|(bx) = (a\|b)x$	CM6	$(\tau x)\|y = x\|y$	TC3
$(ax)\|(by) = (a\|b)(x\|\|y)$	CM7	$x\|(\tau y) = x\|y$	TC4
$(x + y)\|z = x\|z + y\|z$	CM8		
$x\|(y + z) = x\|y + x\|z$	CM9		
		$\partial_H(\tau) = \tau$	DT
		$\tau_I(\tau) = \tau$	TI1
$\partial_H(a) = a$ if $a \notin H$	D1	$\tau_I(a) = a$ if $a \notin I$	TI2
$\partial_H(a) = \delta$ if $a \in H$	D2	$\tau_I(a) = \tau$ if $a \in I$	TI3
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	TI4
$\partial_H(xy) = \partial_H(x).\partial_H(y)$	D4	$\tau_I(xy) = \tau_I(x).\tau_I(y)$	TI5

Table 2.

2. THE MODEL OF FINITE ACYCLIC PROCESS GRAPHS FOR ACP_τ

Let G be the collection of finite acyclic process graphs over A_τ. In order to define the notion of bisimulation on G, we will first introduce the notion of <u>δ-normal</u> process graph. A process graph $g \in G$ is δ-normal if whenever an edge

$$s \xrightarrow{\delta} t$$

occurs in g, then the node s has outdegree 1 and the node t has outdegree 0. In anthropomorphic terminology, let us say that an edge $s \longrightarrow t$ is an <u>ancestor</u> of $s' \longrightarrow t'$ if it is possible to move along edges from t to s'; likewise the latter edge will be called a <u>descendant</u> of the former. Edges having the same begin node are <u>brothers</u>. So, a process graph g is δ-normal if all its δ-edges have no brothers and no descendants.

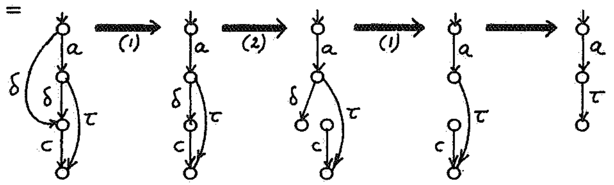
Note that for $g \in G$ the ancestor relation is a partial order on the set of edges of g.

We will now associate to a process graph $g \in G$ a unique g' in δ-normal form, by the following procedure:

(1) <u>nondeterministic δ-removal</u> is the elimination of a δ-edge having at least one brother,

(2) <u>δ-shift</u> of a δ-edge $s \xrightarrow{\delta} t$ in g consists of deleting this edge, creating a fresh node t' and adding the edge $s \xrightarrow{\delta} t'$.

Now it is not hard to see that the procedure of repeatedly applying (in arbitrary order) (1),(2) in g will lead to a unique graph g' which is δ-normal; this g' is the <u>δ-normal form</u> of g. It is understood that pieces of the graph which have become disconnected from the root, are discarded.

<u>Example</u>: g =  = g'.

We can now define bisimulation between process graphs $g_1, g_2 \in G$. First some preliminary notions: a <u>trace</u> σ is a possibly empty finite string over A_τ; thus $\sigma \in A_\tau^*$. With $e(\sigma)$ we denote the trace σ where all τ-steps are erased, e.g. $e(a\tau\tau b\tau c\tau) = abc$.

If $g \in G$, a path $\pi: s_0 \twoheadrightarrow s_k$ in g is a sequence of edges of the form

$$s_0 \longrightarrow s_1 \longrightarrow \cdots\cdots \longrightarrow s_k$$

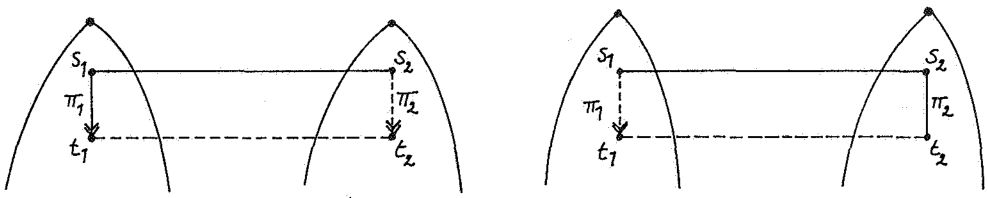
($k \geqslant 0$) where the s_i are nodes of g, the h_i are edges between s_i and s_{i+1}, and each $\ell_i \in A_\tau$ is the label of edge h_i. (The h_i are needed because we work with multigraphs.) The trace <u>trace</u>(π) associated to this path π is just $\ell_0 \ell_1 \ldots \ell_{k-1}$.

2.1. <u>DEFINITION</u>. A <u>bisimulation modulo τ</u> (or <u>τ-bisimulation</u>) between finite acyclic process graphs g_1 and g_2 is a relation R on NODES$(g_1) \times$ NODES(g_2) satisfying the following conditions:

(i) $(\text{ROOT}(g_1), \text{ROOT}(g_2)) \in R$,

(ii) For each pair $(s_1, s_2) \in R$ and for each path $\pi_1: s_1 \twoheadrightarrow t_1$ in g_1 there is a path $\pi_2: s_2 \twoheadrightarrow t_2$ in g_2 such that $(t_1, t_2) \in R$ and $e(\underline{\text{trace}}(\pi_1)) = e(\underline{\text{trace}}(\pi_2))$. (See Figure 1a)

(iii) Likewise for each pair $(s_1, s_2) \in R$ and for each path $\pi_2: s_2 \twoheadrightarrow t_2$ in g_2 there is a path $\pi_1: s_1 \twoheadrightarrow t_1$ in g_1 such that $(t_1, t_2) \in R$ and $e(\underline{\text{trace}}(\pi_1)) = e(\underline{\text{trace}}(\pi_2))$. (See Figure 1b.)



<u>Figure 1.</u> (a) (b)

Let g_1, g_2 be in δ-normal form. Then g_1, g_2 are <u>bisimilar modulo τ</u> (or <u>τ-bisimilar</u>) if there is a τ-bisimulation between g_1, g_2.

<u>Notation</u>: $g_1 \underset{\tau}{\leftrightarrow} g_2$.

Note that for a τ-bisimulation R between g_1, g_2 we have: Domain(R) = NODES(g_1) and Codomain(R) = NODES(g_2). Also note that an equivalent definition is obtained by letting π_1 in 2.1(ii) consist of one edge, likewise π_2 in 2.1(iii).

2.2. <u>DEFINITION</u>. Let $g_1, g_2 \in G$ be in δ-normal form. A <u>rooted bisimulation</u>

modulo τ between g_1, g_2 is a bisimulation modulo τ between g_1, g_2 such that the root of g_1 is not related to a non-root node of g_2, and vice versa.

Notation: $g_1 \underline{\leftrightarrow}_{r,\tau}\, g_2$.

2.3. DEFINITION. Let $g_1, g_2 \in G$ with δ-normal forms g_1' resp. g_2'. Then $g_1 \underline{\leftrightarrow}_{r,\tau}\, g_2$ if $g_1' \underline{\leftrightarrow}_{r,\tau}\, g_2'$.

2.4. EXAMPLES. $a\tau b\delta \underline{\leftrightarrow}_{r,\tau}\, ab\delta$ (Figure 2a);

$ab \underline{\leftrightarrow}_{r,\tau}\, a\tau(\tau b + \tau\tau b)$ (Figure 2b)

$a(\tau b + b) \underline{\leftrightarrow}_{r,\tau}\, ab$ (Figure 2c)

$c(a+b) \underline{\leftrightarrow}_{r,\tau}\, c(\tau(a+b)+a)$ (Figure 2d)

A negative example: see Figure 2e. The heavy line denotes where it is not possible to continue a construction of the bisimulation.

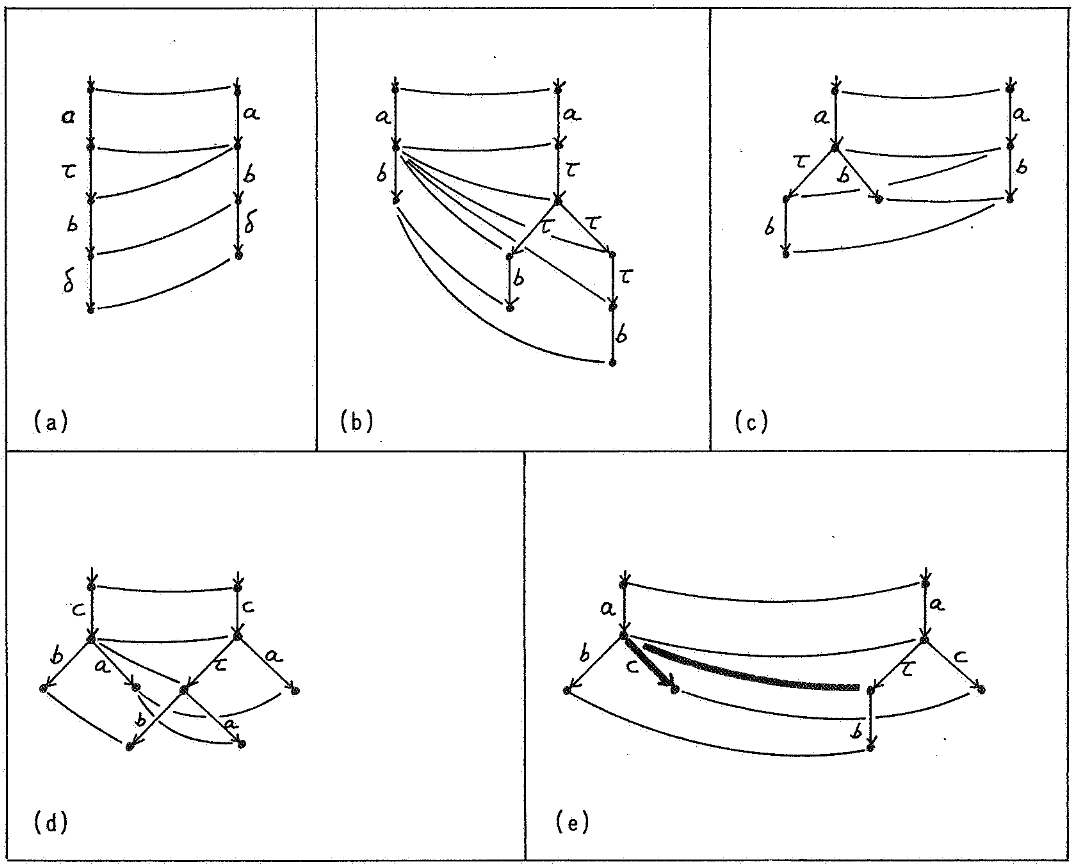


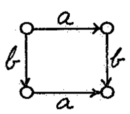
Figure 2.

Since we intend to construct from G a model for ACP_τ, we will now define operations $+, \cdot, \|, \lfloor\!\lfloor, |, \partial_H, \tau_I$ on G. (Cf. [3] where $+, \cdot, \|, \lfloor\!\lfloor$ were defined in the context of the axiom system PA.)

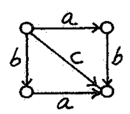
(1) The <u>sum</u> $g_1 + g_2$ is the result of identifying the roots of g_1, g_2.

(2) The <u>product</u> $g_1 \cdot g_2$ is the result of appending g_2 at all end nodes of g_1.

(3) The <u>merge</u> $g_1 \| g_2$ is the 'cartesian product graph' of g_1, g_2, enriched by 'diagonal' edges for nontrivial communication steps, as follows:

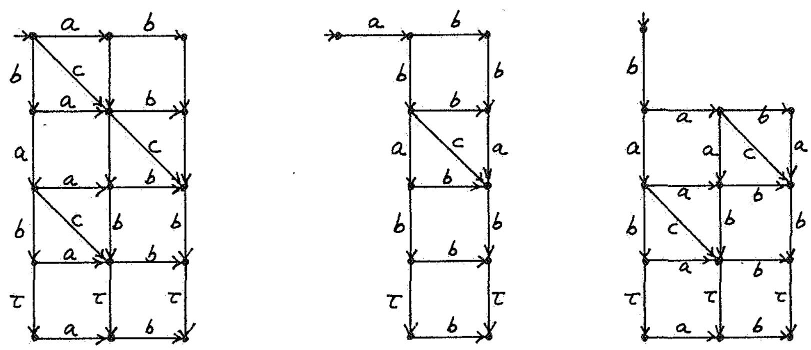
if 

is a subgraph of the cartesian product graph, then the arrow $\circ \xrightarrow{c} \circ$ (where $c = a|b$) is inserted; result:



(Here τ has only trivial communications: $\tau|a = \tau|\tau = \delta$.)

<u>Example</u>. Let $A_\tau = \{a,b,c,\tau,\delta\}$, where the only nontrivial communication is: $a|b = c$. Then, writing ab for the graph $\to\circ \xrightarrow{a} \circ \xrightarrow{b} \circ$, we have: $ab \| bab\tau$ is the process graph as in Figure 3a.



<u>Figure 3.</u> (a) (b) (c)

(4) The <u>left merge</u> $g_1 \lfloor\!\lfloor g_2$ is like $g_1 \| g_2$ but omitting all steps which are not a first step from g_1 or the descendant of such a first step.

<u>Example</u>: in the situation of the previous example we have $ab \lfloor\!\lfloor bab\tau$ as the graph in Figure 3b and $bab\tau \lfloor\!\lfloor ab$ as in Figure 3c .

(Note that we have omitted the diagonal edges labeled with δ, resulting from trivial communications. This is allowed in view of our preference of δ-normal graphs. Indeed, a 'diagonal' δ-edge can always be omitted by (1) of the δ-normalization procedure.)

(5) The communication merge $g_1|g_2$ is harder to define since it is in general not, as $g_1 \mathbin{⌊\!⌊} g_2$ is, a subgraph of $g_1 \| g_2$. The reason behind the definition can be understood by considering e.g. $\tau\tau ax|\tau\tau\tau by$ and evaluating this term according to the axioms of ACP :

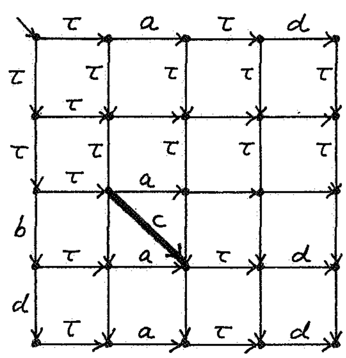
$\tau\tau ax|\tau\tau\tau by = ax|by = (a|b).(x\|y)$.

We define:

$$g_1|g_2 = \sum\{(t \longrightarrow s).(g_1\|g_2)_s \mid t \longrightarrow s \text{ is a maximal communication step in } g_1\|g_2 \text{ such that } t \text{ can be reached from the root via a sequence of } \tau\text{-steps}\}.$$

Here 'maximal' refers to the p.o. given by the ancestor relation. The sequence of τ-steps may be empty. Further, $(g)_s$ denotes the subgraph of g with root s.

Example. (i) Let $g_1 = \tau a\tau d$, $g_2 = \tau\tau bd$. Let $a|b = c$ be the only nontrivial communication. Then $g_1\|g_2$ is as in Figure 4(a) and $g_1|g_2$ as in Figure 4(b):



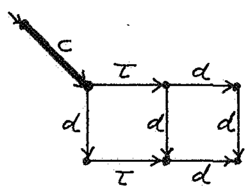


Figure 4. (a) (b)

Here the heavily drawn edge $\circ \xrightarrow{c} \circ$ is an edge $t \longrightarrow s$ as in the definition of $g_1|g_2$.

(ii) Let g_1 be $\rightarrow\circ \overset{b}{\underset{\tau}{\rightrightarrows}} \circ \xrightarrow{a} \circ$ and g_2: $\rightarrow\circ \overset{a}{\underset{\tau}{\rightrightarrows}} \circ \xrightarrow{b} \circ$, where the only non-trivial communications are $a|a = a°$ and $b|b = b°$. Then $g_1 \| g_2$ and $g_1 | g_2$ are as in Figures 5(a) resp. (b):

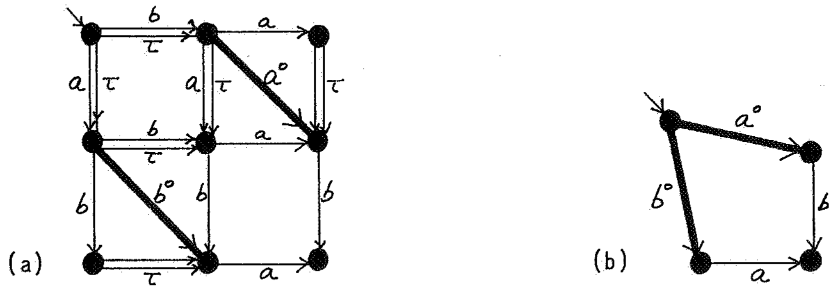


Figure 5.

Using ACP_τ we calculate with terms corresponding to g_1, g_2:

$$(ba + \tau a) \mid (ab + \tau b) = ba|ab + ba|\tau b + \tau a|ab + \tau a|\tau b =$$

$$(b|a).(a\|b) + ba|b + a|ab + a|b = \delta + b°a + a°b + \delta = b°a + a°b.$$

(6) The definition of the operators ∂_H, τ_I on process graphs $g \in G$ is easy: they merely rename some atoms (labels at the edges) into δ resp. τ.

This ends the definition of the structure $\mathcal{G} = G(+,.,\|,\lfloor\!\lfloor,|,\partial_H,\tau_I)$. The domain of process graphs $\mathcal{G}$ is itself not yet a model of ACP (e.g. $\mathcal{G} \not\models x + x = x$). However:

2.5. <u>THEOREM</u>. (i) <u>Rooted τ-bisimulation ($\leftrightarrows_{r,\tau}$) is a congruence on $\mathcal{G}$</u>.
(ii) <u>$\mathcal{G}/\leftrightarrows_{r,\tau}$ is a model of ACP_τ</u>.

<u>PROOF</u>. (i) Let $g, g', h, h' \in G$. We want to show that

$$g \leftrightarrows_{r,\tau} g' \ \&\ h \leftrightarrows_{r,\tau} h' \implies g\|h \leftrightarrows_{r,\tau} g'\|h'$$

and likewise for the other operators. Only the cases $\|, \lfloor\!\lfloor, |$ are interesting and we start with $\|$.

Suppose, then, that S is a r,τ-bisimulation between g, g' and T is a r,τ-bisimulation between h, h'. Let s be a typical node of g, s' of g', t of h and t' of h'. Then we define the following relation $S \times T$ between the node sets of $g\|h$ and $g'\|h'$:

$$((s,t),(s',t')) \in S \times T \iff (s,s') \in S \ \&\ (t,t') \in T.$$

We claim that $S \times T$ is a r,τ-bisimulation between $g\|h$ and $g'\|h'$.

<u>Proof of the claim</u>.

(1) Let $(s_1,t_1) \xrightarrow{u} (s_1,t_2)$ be a "horizontal step" in $g||h$, where $u \in A_\tau$. Let $((s_1,t_1),(s_1',t_1')) \in S \times T$. Then $t_1 \xrightarrow{u} t_2$ in h and $(t_1,t_1') \in T$. Hence a path as in the definition of bisimulation can be found whose trace is externally equivalent to u and whose end point bisimulates with t_2. This path can be 'lifted' to $g||h$.

(2) Likewise for a "vertical step" in $g||h$.

(3) $(s_1,t_1) \xrightarrow{c} (s_2,t_2)$ is a "diagonal step" (a communication step) in $g||h$, and $((s_1,t_1),(s_1',t_1')) \in S \times T$. Now a path as required can be found from the data $(s_1,s_1') \in S$ and $(t_1,t_1') \in T$ and an inspection of Figure 6:

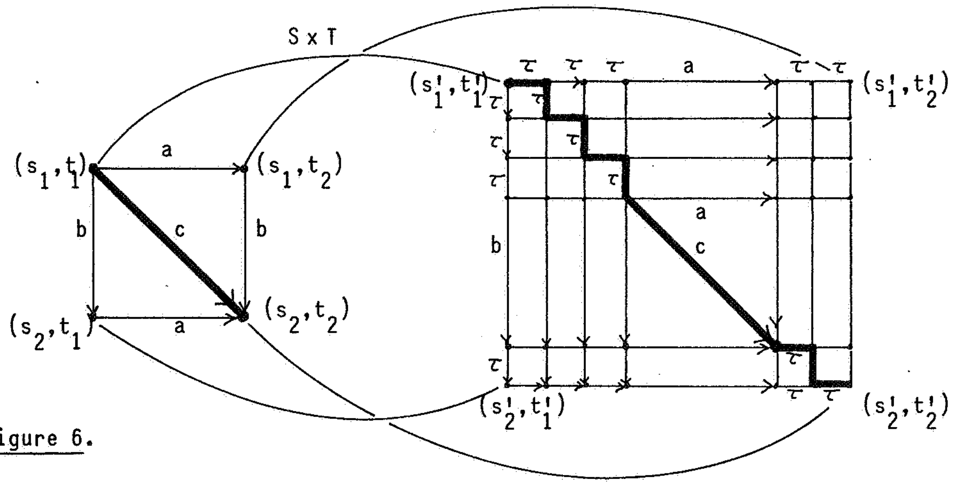


Figure 6.

The case of $\lfloor\!\lfloor$ is easy since $g \lfloor\!\lfloor h$ is a subgraph of $g||h$.

The case of $|$: we use the same notation as above. To prove:

$$g|h \underset{r,\tau}{\leftrightarrows} g'|h'.$$

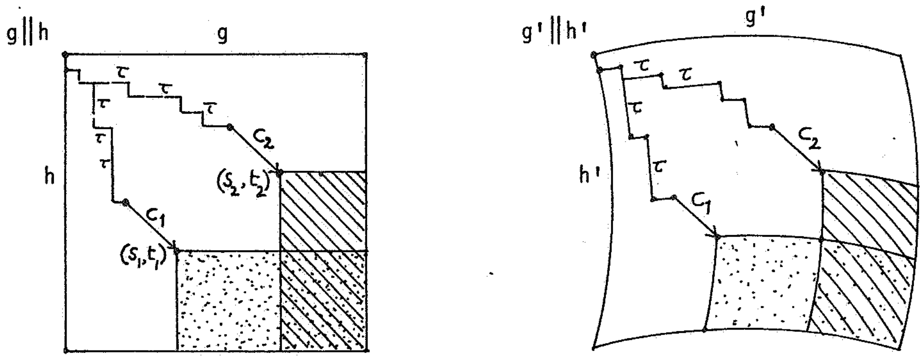


Figure 7.

An r,τ-bisimulation between $g|h$ and $g'|h'$ can now be constructed as follows from $S \times T$. The graph $g|h$ is now the sum of the $c_i\cdot(g||h)_{(s_i,t_i)}$ $(i=1,2)$ as in the definition of $|$ and as indicated in Figure 7 (a).

For the sake of clarity, we will formally distinguish the "diagonal" edges from the other ones; this can be done by a suitable renaming of the alphabet and adapting the communication function. Thus, if $a|b=c$, we adopt a fresh symbol $\underline{c}$ and postulate $a|b=\underline{c}$. Now the underlined symbols do not occur in g,h which makes it possible to speak in a formal way about "diagonal" steps. Note that the bisimulation $S\times T$ is also a bisimulation when diagonal steps are marked as such.

Now given a summand $p = c_i\cdot(g||h)_{(s_i,t_i)}$ of $g|h$, we can find via $S\times T$ a corresponding summand $p' = c_i\cdot(g'||h')_{(s_i',t_i')}$. It is easy to see that the step c_i in $g'||h'$ is also maximal in the sense of the definition of $|$. Clearly p bisimulates with p', via the restriction of S T to the appropriate area. In this way we find that $g|h$ bisimulates with $g'|h'$.

(ii) The proof that $\mathcal{G}/{\leftrightarrows_{r,\tau}}$ is a model of ACP is tedious, routine, and omitted. ☐

We will now analyse $\leftrightarrows_{r,\tau}$ into an equivalence generated by certain elementary graph reductions. This is done in [3] for τ-bisimulation (without the condition 'rooted') and in the absence of δ; these results will be the basis for the sequel. We repeat from [3] the main definitions.

2.6. DEFINITION. Let $g \in G$.

(i) A subgraph g' of g consists of an arbitrary subset of the set of edges of g (plus their labels $\in A_\tau$) together with the nodes belonging to these edges.

(ii) Let $s \in NODES(g)$. Then $(g)_s$ is the subgraph of g consisting of all nodes and edges which are accessible from s (including s, the root of $(g)_s$). We will call $(g)_s$ a full subgraph.

(iii) An arc in g is a subgraph of the form as in Figure 8(a), where $u \in A_\tau$. The u-edge at the left is called the primary edge of the arc. If in Figure 8 (a) $n=m=0$ the arc has the form as in Figure 8 (b) and is called of type I. If $n+m = 1$ the arc has the form as in Figure 8 (c) or (d) and is called of type II resp. III. Arcs of type I,II,III are called elementary arcs.

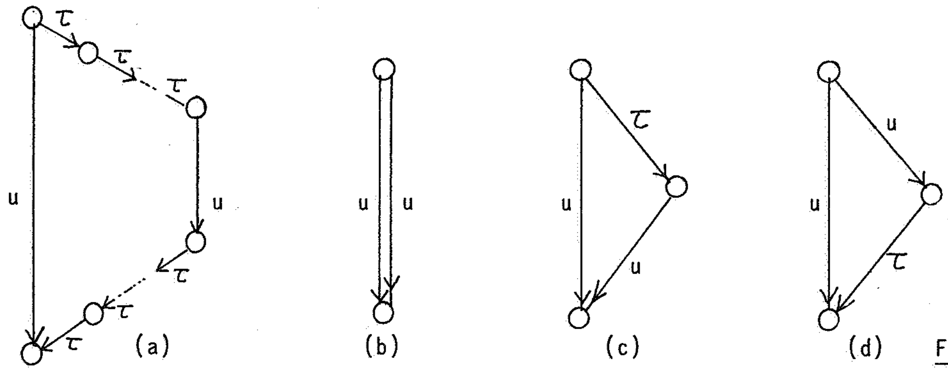


Figure 8.

2.7. DEFINITION. On G we define the following reduction procedures:

[i] Sharing. Let $g \in G$ contain nodes s_1, s_2 such that $(g)_{s_1}$ is isomorphic to $(g)_{s_2}$. Then g reduces to g' where s_1, s_2 are identified.

[ii] Removal of a non-initial deterministic τ-step.
If $s_1 \xrightarrow{\tau} s_2$ occurs in g and the outdegree of s_1 is one (so the displayed τ-step has no brothers), and if moreover s_1 is not the root of g, then the nodes s_1, s_2 may be identified after removal of the τ-step.

[iii] Arc reduction. In an arc the primary edge may be deleted. The arc reduction is called of type I,II,III if the arc is of that type. Such arc reductions are also called elementary.
So the subgraph as in Figure 9(a) may be replaced by that in Figure 9(b):

Figure 9.


[iv] Nondeterministic δ-removal, as explained in the beginning of this section.
[v] δ-shift; also defined above.

If none of the reduction possibilities in [i]-[v] applies to g, then we call g a normal process graph.

Notation. If g reduces to g' by one application of [i]-[v], we write $g \Longrightarrow g'$. The transitive reflexive closure of $\Longrightarrow$ is denoted by $\Rrightarrow$.

2.8. EXAMPLE.

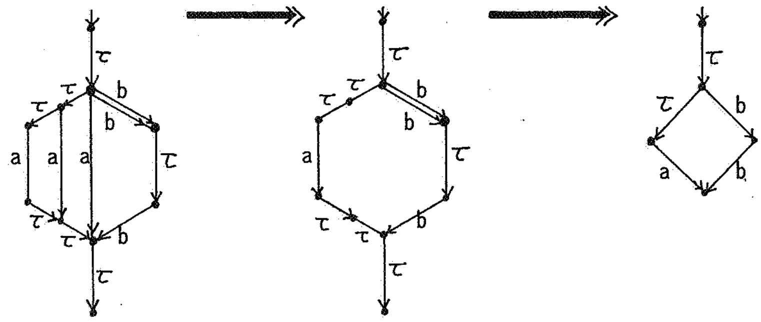


Figure 10.

The following fact is trivial:

2.9. PROPOSITION. Every process graph reduction $g_1 \Rightarrow g_2 \Rightarrow \ldots$ must terminate eventually.

Without the routine proof we state the 'soundness' of the reduction procedure $\Rightarrow$ w.r.t. $\underline{\leftrightarrow}_{r,\tau}$:

2.10. LEMMA. Let $g_1, g_2 \in G$. Then $g_1 \Rightarrow g_2$ implies $g_1 \underline{\leftrightarrow}_{r,\tau}\ g_2$.

2.11. DEFINITION. (i) Let $g \in G$ be in δ-normal form. Let R be an r,τ-bisimulation between g and itself. Then R is called an autobisimulation of g.
(ii) g is rigid if it can only be in autobisimulation with itself via the identity relation.

2.11.1. EXAMPLE. The following process graph is not rigid since it admits the displayed nontrivial autobisimulation:

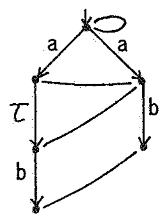


Figure 11.

2.12. THEOREM. (i) Normal process graphs are rigid.
(ii) If g_1, g_2 are normal process graphs and $g_1 \underline{\leftrightarrow}_{r,\tau}\ g_2$, then g_1 and g_2 must be identical.

PROOF. The theorem is a simple corollary of the analogous Theorem 8.1.9 in [3], where 'normal', 'rigid' are defined w.r.t $\underline{\leftrightarrow}_{\tau}$ (without the condition 'rooted') and in the absence of δ. The present graph reductions [i]-[v] differ from those in [3] since there [iv],[v] are absent and in [ii] the τ-step may be an initial one.

Proof of (ii): suppose g_1, g_2 are normal and $g_1 \overset{\leftrightarrow}{=}_{r,\tau} g_2$.

Case (1). g_1, g_2 are also 'normal' in the sense of [3]. Then since $g_1 \overset{\leftrightarrow}{=}_{r,\tau} g_2$ implies $g_1 \overset{\leftrightarrow}{=}_{\tau} g_2$, an application of Theorem 8.1.9 in [3] yields the identity of g_1, g_2.

Case (2). If g_1, g_2 are normal but not 'normal' as in [3], one of them, say g_1, must start with a deterministic τ-step: i.e. $g_1 = \tau g_1'$. Then since $g_1 \overset{\leftrightarrow}{=}_{r,\tau} g_2$, also $g_2 = \tau g_2'$. Moreover, g_1', g_2' must be 'normal' as in [3]. Also $g_1' \overset{\leftrightarrow}{=}_{r,\tau} g_2'$, hence $g_1' \overset{\leftrightarrow}{=}_{\tau} g_2'$. By Theorem 8.1.9 in [3], we have $g_1' = g_2'$. Therefore $g_1 = g_2$.

Proof of (i): similar. □

2.13. COROLLARY. *Let $g_1, g_2 \in G$. Then the following are equivalent:*

(i) $g_1 \overset{\leftrightarrow}{=}_{r,\tau} g_2$

(ii) g_1, g_2 *reduce (by [i]-[v]) to the same normal graph*

(iii) g_1, g_2 *are convertible via applications of [i]-[v].*

PROOF. Suppose (i). Reduce g_1, g_2 to normal g_1', g_2'; this is possible by Proposition 2.9. Since reduction $\twoheadrightarrow$ is sound w.r.t $\overset{\leftrightarrow}{=}_{r,\tau}$, also $g_1' \overset{\leftrightarrow}{=}_{r,\tau} g_2'$. By Theorem 2.12(ii) it follows that g_1' and g_2' are identical. Hence (ii). From (ii) we have (iii) trivially. From (iii), since reduction is sound, we have again (i). □

2.14. REMARK. As a further corollary (which we do not need here) one obtains the confluency of the graph reductions [i]-[v]. This follows immediately from the termination property of the graph reductions (Proposition 2.9), together with Lemma 2.10 and Theorem 2.12(ii).

2.15. COROLLARY. *Let $g_1, g_2 \in G$. Then $g_1 \overset{\leftrightarrow}{=}_{r,\tau} g_2$ iff g_1, g_2 are convertible by means of the graph reductions [i],[ii],[iv],[v] and elementary arc reductions [iii]I,[iii]II,[iii]III.*

PROOF. Every arc can be filled up with elementary arcs. □

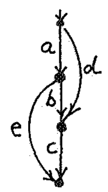
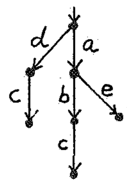
In the sequel when closed terms in the signature $(+,\cdot,a\in A_\tau)$ are mentioned, we will always mean terms modulo the basic congruence given by the axioms A1,2,5 in Table 2 (associativity of $+,\cdot$ and commutativity of $+$). To such terms we will refer as '$+,\cdot$-terms' or as '<u>basic terms</u>'.

2.16. <u>DEFINITION</u>. Let t be a basic term.

(i) Then $[t]$ denotes the interpretation of t in $\mathcal{G}$; so $[t]$ is a process graph.

(ii) $[t]$ denotes the interpretation of t in $\mathcal{G}/{\leftrightarrows_{r,\tau}}$; so $[t]$ is a process graph modulo r,τ-bisimulation.

(iii) Let $g\in G$. Let g' be the process tree obtained from g by 'unraveling' the shared subgraphs. Then $\{g\}$ is the basic term corresponding to the tree g'.

<u>Example</u>. If g is  then g' is  and $\{g\} = dc + a(bc + e)$.

2.17. <u>PROPOSITION</u>. <u>Let $g_1,g_2\in G$ and suppose $g_1 \Longrightarrow g_2$ via an elementary graph reduction [i],[ii],[iiiI,II,III],[iv],[v]. Then the basic terms $\{g_1\}$ and $\{g_2\}$ can be proved equal using the A-axioms (about $+,\cdot,\delta$) in Table 2, A1-7, and the τ-laws T1-3</u>. (See Figure 12)

$$\begin{tikzcd}
g_1 \arrow[r, Rightarrow, "\text{elementary graph reduction step}"] \arrow[d, "\{\,\}"'] & g_2 \arrow[d, "\{\,\}"] \\
t_1 \arrow[r, equal, "\exists", "\text{A1-7, T1-3}"'] & t_2
\end{tikzcd}$$

<u>Figure 12</u>.

<u>PROOF</u>. In case [i], $t_1 \equiv t_2$. Case [ii] translates into an application of T1 (or several such). Case [iiiI]: removal of a double edge. This translates into applications of $x + x = x$ (A3).

Case [iiiII] translates to terms as an application of $\tau(x+y) + x = \tau(x+y)$, where $x = uz$ (see Figure 13a), or, if y is empty, $\tau x + x = \tau x$ (T2). The former equation follows from T2 and A3:

$$\tau(x+y)+x = \tau(x+y)+x+y+x = \tau(x+y)+x+y = \tau(x+y).$$

Case [iiiIII] translates to terms as an application of

$$u(\tau z+y) = u(\tau z+y)+uz \quad (u \in A_\tau).$$

(See Figure 13b) The case that $u=\tau$ follows from T2; the case that $u \neq \tau$ is just the third τ-law T3; for z or y empty an application of T1 is needed. □

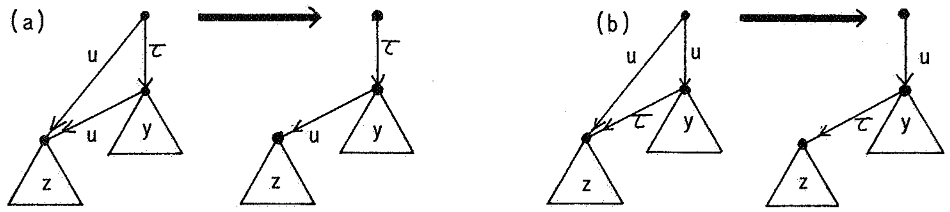


Figure 13.

Now we can prove an important fact:

2.18. LEMMA. Suppose t,s are basic terms. Then:

$$\mathcal{G}/{\leftrightarrows_{r,\tau}} \models t=s \implies \text{A1-7, T1-3} \vdash t=s.$$

PROOF. Suppose $\mathcal{G}/{\leftrightarrows_{r,\tau}} \models t=s$. Then $[t] \leftrightarrows_{r,\tau} [s]$. By Corollary 2.15, the graphs [t], [s] are convertible via elementary graph reductions:

$$[t] \equiv g_0 \text{—} g_1 \text{—} \ldots \text{—} g_n \equiv [s].$$

Now Proposition 2.17 states that

$$\text{A1-7, T1-3} \vdash \{[t]\} = \{g_1\} = \ldots = \{g_n\} = \{[s]\}.$$

Since A1-7 $\vdash \{[t]\} = t$ and likewise for s, we have A1-7,T1-3 $\vdash t=s$. □

By a similar method (essentially by leaving out all reference to τ) one proves

2.19. LEMMA. Suppose t,s are basic terms not containing τ. Then:

$$\mathcal{G}/{\leftrightarrows_{r,\tau}} \models t=s \implies \text{A1-7} \vdash t=s.$$

2.20. ELIMINATION THEOREM. Let t be a closed term in the signature of ACP_τ. Then, using the axioms of ACP_τ except A1-7 and the τ-laws T1-3 as rewrite rules from left to right, t can be rewritten to a basic term t'.

PROOF. See Appendix I. □

Combining the previous results we now have, writing AT for the set of axioms A1-7,T1-3:

2.21. LEMMA. (i)

$$\begin{tikzcd} t_1 \arrow[rr, equal, "ACP_\tau"] \arrow[d, twoheadrightarrow, "ACP_\tau\text{-}AT"'] & & t_2 \arrow[d, twoheadrightarrow, "ACP_\tau\text{-}AT"] \\ t_3 \arrow[rr, equal, "\exists", "AT"'] & & t_4 \end{tikzcd}$$

Figure 14.

I.e. if $ACP_\tau \vdash t_1 = t_2$, then t_1 and t_2 can be reduced by means of the rewrite rules (from left to right) associated to the axioms in ACP_τ-AT to basic terms t_3, t_4 which are convertible via the AT-axioms.

(ii) Every term t can be proved equal in ACP_τ to a basic term t'; moreover, t' is unique modulo AT.

PROOF. (i) Suppose $ACP_\tau \vdash t_1 = t_2$. By the Elimination Theorem 2.20 we can rewrite t_1, t_2 to resp. basic terms t_3, t_4 using the axioms in ACP_τ-AT as rewrite rules. By the fact that $\mathcal{G}/\underline{\leftrightarrow}_{r,\tau}$ is a model of ACP_τ we have $\mathcal{G}/\underline{\leftrightarrow}_{r,\tau} \models t_3 = t_4$. Hence (Lemma 2.18) $AT \vdash t_3 = t_4$.
(ii) Immediate from (i). □

2.22. EXAMPLES. The following examples illustrate Lemma 2.21(i):

(i)

$$\begin{tikzcd} (\tau a + a)|b \arrow[r, equal] \arrow[d] & \tau a|b \arrow[dd] \\ \tau a|b + a|b \arrow[d] & \\ a|b + a|b \arrow[r, equal] & a|b \end{tikzcd}$$

(ii)

$$\begin{tikzcd} a\tau \mathbin{\lfloor\!\lfloor} b \arrow[r, equal] \arrow[d] & a \mathbin{\lfloor\!\lfloor} b \arrow[ddd] \\ a(\tau \| b) \arrow[d] & \\ a(\tau \mathbin{\lfloor\!\lfloor} b + b \mathbin{\lfloor\!\lfloor} \tau + \tau|b) \arrow[d] & \\ a(\tau b + b\tau + \delta) = a(\tau b + b\tau) = a(\tau b + b) = a\tau b & = ab \end{tikzcd}$$

(iii)

$$\begin{tikzcd}
(\tau a + a) \mathbin{|\!\lfloor} b \arrow[r, equal] \arrow[d] & \tau a \mathbin{|\!\lfloor} b \arrow[d] \\
\tau a \mathbin{|\!\lfloor} b + a \mathbin{|\!\lfloor} b \arrow[d] & \tau(a \| b) \arrow[ddd, twoheadrightarrow] \\
\tau(a\|b) + a \mathbin{|\!\lfloor} b \arrow[d] & \\
\tau(a \mathbin{|\!\lfloor} b + b \mathbin{|\!\lfloor} a + a|b) + a \mathbin{|\!\lfloor} b \arrow[d, twoheadrightarrow] & \\
\tau(ab + ba + a|b) + ab \arrow[r, equal, "(*)"'] & \tau(ab + ba + a|b)
\end{tikzcd}$$

Here (*) is an instance of the (from AT) derivable rule $\tau(x+y)+x = \tau(x+y)$.

As a further corollary we have:

2.23. <u>THEOREM</u>. (i) <u>$\mathcal{G}/\underline{\leftrightarrow}_{r,\tau}$ is isomorphic to A_τ^ω, the initial algebra of ACP_τ</u>.

(ii) <u>ACP_τ is conservative over ACP (the latter over the alphabet A)</u>.
<u>I.e., for τ-less terms</u> t_1, t_2:

$$ACP_\tau \vdash t_1 = t_2 \;\Rightarrow\; ACP \vdash t_1 = t_2.$$

<u>PROOF</u>. (i) We have to prove:

$$\mathcal{G}/\underline{\leftrightarrow}_{r,\tau} \models s = t \;\Leftrightarrow\; ACP_\tau \vdash s = t.$$

($\Leftarrow$) is Theorem 2.5(ii). For ($\Rightarrow$), suppose $\mathcal{G}/\underline{\leftrightarrow}_{r,\tau} \models s = t$. Then also $\mathcal{G}/\underline{\leftrightarrow}_{r,\tau} \models s' = t'$ for some basic terms s', t' such that $ACP_\tau \vdash s = s', t = t'$. The result now follows by Lemma 2.18.

(ii): Suppose t_1, t_2 are closed terms in the signature of ACP (so τ-less and τ_I-less), and suppose $ACP_\tau \vdash t_1 = t_2$. Let t_3, t_4 be basic terms such that $ACP_\tau \vdash t_1 = t_3,\ t_2 = t_4$. Since t_3, t_4 can be obtained by rewrite rules ACP_τ-AT, we have $ACP \vdash t_1 = t_3,\ t_2 = t_4$. Now by Lemma 2.19, A1-7 $\vdash t_3 = t_4$. Hence $ACP \vdash t_1 = t_2$. □

3. THE EXPANSION THEOREM FOR ACP_τ

The Expansion Theorem is an important algebraic tool since it helps in breaking down a merge expression $x_1 \| x_2 \| \ldots \| x_k$. For CCS, an Expansion Theorem is proved in MILNER [12]. For ACP (i.e ACP_τ without τ) the analogous theorem is proved in BERGSTRA & TUCKER [5]. As an example we mention the Expansion Theorem for ACP in the case $k=3$:

$$x\|y\|z = x \mathbin{\lfloor\!\lfloor} (y\|z) + y \mathbin{\lfloor\!\lfloor} (z\|x) + z \mathbin{\lfloor\!\lfloor} (x\|y) + (y|z) \mathbin{\lfloor\!\lfloor} x + (z|x) \mathbin{\lfloor\!\lfloor} y + (x|y) \mathbin{\lfloor\!\lfloor} z.$$

In [5], the Expansion Theorem is proved by a straightforward induction on k starting from the assumptions:

(a) the <u>handshaking axiom</u> $x|y|z = \delta$ (i.e. communications are binary),

(b) the <u>axioms of standard concurrency</u> for ACP:

$$\boxed{\begin{aligned}
(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z &= x \| (y\|z)\\
(x|y) \mathbin{\lfloor\!\lfloor} z &= x|(y \mathbin{\lfloor\!\lfloor} z)\\
x|y &= y|x\\
x\|y &= y\|x\\
x|(y|z) &= (x|y)|z\\
x\|(y\|z) &= (x\|y)\|z
\end{aligned}}$$

<u>Table 3.</u>

The standard concurrency axioms are fulfilled in the main models of ACP, to wit the term model (initial algebra) A_ω of ACP, the projective limit model A^∞ and the graph model $\mathbb{A}^\infty$ (see [4]).

For ACP_τ this is no longer true; all axioms of standard concurrency hold in the initial algebra A_τ^ω of ACP_τ except the second one.

<u>Example</u>: $(a|\tau b) \mathbin{\lfloor\!\lfloor} c = (a|b)c$ and $a|(\tau b \mathbin{\lfloor\!\lfloor} c) = (a|b)c + (a|c)b + a|b|c$.

For a proof of the validity of some of the axioms of standard concurrency in A_τ^ω, see Appendix II.

Fortunately, the Expansion Theorem carries over from ACP to ACP_τ in exactly the same form. This is what we will prove in this section. The underlying intuition is that $\|$ and $\lfloor\!\lfloor$ behave in ACP_τ just like in ACP, with the convention that τ cannot communicate. For '$|$' the same is true if its arguments x,y are 'saturated' in the sense that they have been maximally exposed to the rewrite rule associated to T2: $\tau x \longrightarrow \tau x + x$. As an example, consider $\tau a|b$. Evaluated according to ACP, we have

$$\tau a|b = (\tau|b)a = \delta a = \delta.$$

However, according to ACP_τ:

$$\tau a|b = a|b,$$

which may be different from δ. Now suppose that τa is made 'saturated' in the above sense, i.e. replaced by $\tau a + a$. Then also by ACP:

$$(\tau a + a)|b = \tau a|b + a|b = (\tau|b)a + a|b = \delta + a|b = a|b,$$

just as in ACP_τ.

The proof below of the Expansion Theorem will also entail the associativity of $\|$. Nevertheless, we have given in Appendix a totally different proof of the associativity of $\|$ in A_τ^ω, by means of an induction to term complexity. This is done, because the latter proof yields some useful identities (some of the axioms of standard concurrency) and for the curious fact that the proof requires an application of the third τ-law (T3). (In computations with and applications of ACP_τ the first two τ-laws turn up frequently; this seems not to be the case for the third τ-law.)

3.1. <u>DEFINITION</u>. T is the set of basic terms in normal form w.r.t. the rewrite rule associated to A4: $(x + y)z \longrightarrow xz + yz$. (This means that if $t \in T$, then $[t]$, the interpretation of t in the domain of process graphs G in Section 2, is a process <u>tree</u>.)

3.2. <u>NOTATION</u>. Let $s,t \in T$. We write $s \sqsubseteq t$, if s is a summand of s, i.e. if $t = s$ or $t = s + r$ for some r.

<u>Example</u>: $a(\tau b + c) \sqsubseteq a(\tau b + c) + ab$.

3.3. <u>DEFINITION</u>. Let $x \in T$. Then x is <u>saturated</u> if:

$$\tau y \sqsubseteq x \;\Rightarrow\; y \sqsubseteq x.$$

<u>Example</u>: (i) $b + \tau a$ is not saturated but becomes so after an application of

the τ-law T2: $b+\tau a+a$.

(ii) $b+\tau(a+\tau c)+a+\tau c+c$ is saturated.

3.4. PROPOSITION. Let $x \in T$. Then there exists a saturated $y \in T$ such that $ACP_\tau \vdash x=y$ (in fact, even $T2 \vdash x=y$).

3.5. NOTATION. We will denote by $\bar{x}$ a saturated y as in Proposition 3.4. For definiteness, we take y of minimal length. So, e.g., $\overline{b+\tau a} = b+\tau a+a$.

The next proposition says that a merge in ACP_τ (anyway in its initial algebra A_τ^ω) can be carried out by treating the atom τ as if it were an 'ordinary', non-communicating atom. Formally, this can be expressed by extending the alphabet with a fresh symbol t (acting as a stand-in for τ) which does not communicate, replacing all τ's in a merge by t and after evaluating the merge restoring the τ's by means of the operator $\tau_{\{t\}}$. The same is true for $⌊\!⌊$; for $|$ it is true under the condition that the arguments are saturated. Thus:

3.6. PROPOSITION. Let $x,y \in T$ be terms over the alphabet A_τ. Let $t \notin A_\tau$ and extend the communication function on A_τ to $(A \cup \{t\})_\tau$ such that t does not communicate. Further, let x^t be the term resulting from replacing all occurrences of τ by t. Then:

(i) $ACP_\tau \vdash x \| y = \tau_{\{t\}}(x^t \| y^t)$

(ii) $ACP_\tau \vdash x ⌊\!⌊ y = \tau_{\{t\}}(x^t ⌊\!⌊ y^t)$

(iii) $ACP_\tau \vdash x | y = \tau_{\{t\}}(x^t | y^t)$

PROOF. (i) Let $x = (\tau) + \sum a_i + \sum b_j x_j' + \sum \tau x_k''$, and

$$y = (\tau) + \sum c_\ell + \sum d_m y_m' + \sum \tau y_p''$$

where $a_i, b_j, c_\ell, d_m \in A$.

Then $x \| y = x ⌊\!⌊ y + y ⌊\!⌊ x + x|y =$

$$\left.\begin{array}{llll} (\tau y) & + \sum a_i y & + \sum b_j(x_j' \| y) & + \sum \tau(x_k'' \| y) \\ (\tau x) & + \sum c_\ell x & + \sum d_m(y_m' \| x) & + \sum \tau(y_p'' \| x) \end{array}\right\} \begin{array}{l} + \\ + \end{array}$$

$$
\begin{array}{lllll}
(\tau|\tau) & + \; (\sum \tau|c_\ell) & + \; (\sum \tau|d_m y'_m) & + \; (\sum \tau|\tau y''_p) & + \\
(\sum a_i|\tau) & + \; \sum a_i|c_\ell & + \; \sum a_i|d_m y'_m & + \; \boxed{\sum a_i|\tau y''_p} & + \\
(\sum b_j x'_j|\tau) & + \; \sum b_j x'_j|c_\ell & + \; \sum b_j x'_j|d_m y'_m & + \; \boxed{\sum b_j x'_j|\tau y''_p} & + \\
(\sum \tau x''_k|\tau) & + \; \boxed{\sum \tau x''_k|c_\ell} & + \; \boxed{\sum \tau x''_k|d_m y'_m} & + \; \boxed{\sum \tau x''_k|\tau y''_p} & .
\end{array}
$$

Here the five enclosed summands can be skipped, in view of the following

Claim: $x' \sqsubseteq x$ & $y' \sqsubseteq y \implies x'|y' \sqsubseteq x|y \sqsubseteq \tau(x\|y)$.

Proof of the claim. If $x' \sqsubseteq x$, $y' \sqsubseteq y$ then by the linearity laws CM8,9 for '$|$' at once: $x'|y' \sqsubseteq x|y$. Further, $x|y \sqsubseteq \tau(x\|y)$ follows since

$$ACP_\tau \vdash \tau(x\|y) = \tau(x ⌊\!⌊ y + y ⌊\!⌊ x + x|y) =$$

$$\tau(x ⌊\!⌊ y + y ⌊\!⌊ x + x|y) + x|y.$$

So, e.g. the summand $\sum a_i|\tau y''_p = \sum a_i|y''_p \sqsubseteq \sum \tau(y''_p\|x)$ (since $a_i \sqsubseteq x$); likewise the other four enclosed summands can be shown to be summands of non-enclosed summands. On the other hand, the five corresponding summands in $\tau_{\{t\}}(x^t\|y^t)$ are equal to δ, since t does not communicate. The remaining summands pose no problem, e.g.:

$$\sum b_j(x'\|y) = \tau_{\{t\}} \sum b_j(x'^t_j\|y^t)$$

follows by

$$\tau_{\{t\}} \sum b_j(x'^t_j\|y^t) = \sum b_j \tau_{\{t\}}(x'^t_j\|y^t)$$

and the induction hypothesis

$$x'_j\|y = \tau_{\{t\}}(x'^t_j\|y^t)$$

(induction on the sum of the term complexities).

(ii) The case of $⌊\!⌊$ is similar to that of $⌊\!⌊$.

(iii) It is easy to show that a saturated term $\bar{x} \in T$ can be decomposed as follows:

$$\bar{x} = (\tau) + \sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j y_j + \sum_{k=1}^{\ell} \tau \overline{x}_k$$

where $a_i, b_j \in A$, $n, m, \ell \geqslant 0$ and the $\bar{x}_k$ are again saturated. Note that the length of $\bar{x}_k$ is less than that of $\bar{x}$. We will use this for an induction on the lengths of $\bar{x}, \bar{y}$ in the statement to prove.

We consider a typical example; the general proof involves only greater notational complexity. Let

$$\bar{x} = a + bx_1 + \tau\bar{x}_2 + \bar{x}_2$$

$$\bar{y} = \tau + c + dy_1 + \tau\bar{y}_2 + \bar{y}_2.$$

Then

$$\bar{x}|\bar{y} =$$

$$\begin{array}{llllllllll}
a|\tau & + & a|c & + & a|dy_1 & + & \boxed{a|\tau\bar{y}_2} & + & a|\bar{y}_2 & + \\
bx_1|\tau & + & bx_1|c & + & bx_1|dy_1 & + & \boxed{bx_1|\tau\bar{y}_2} & + & bx_1|\bar{y}_2 & + \\
\boxed{\tau\bar{x}_2|\tau} & + & \boxed{\tau\bar{x}_2|c} & + & \boxed{\tau\bar{x}_2|dy_1} & + & \boxed{\tau\bar{x}_2|\tau\bar{y}_2} & + & \boxed{\tau\bar{x}_2|\bar{y}_2} & + \\
\bar{x}_2|\tau & + & \bar{x}_2|c & + & \bar{x}_2|dy_1 & + & \boxed{\bar{x}_2|\tau\bar{y}_2} & + & \bar{x}_2|\bar{y}_2 & .
\end{array}$$

Note that the enclosed summands can be skipped, since (by virtue of the saturation requirement) they are equal to other summands: e.g. $a|\tau\bar{y}_2 = a|\bar{y}_2$ (by axiom IC4), $bx_1|\tau\bar{y}_2 = bx_1|\bar{y}_2$. Now these are just the terms which are 'lost' when evaluating $\tau_{\{t\}}(\bar{x}^t|\bar{y}^t)$ (since t does not communicate). Namely:

$$\bar{x}^t|\bar{y}^t =$$

$$\begin{array}{llllllllll}
a|t & + & a|c & + & a|dy_1^t & + & \delta & + & a|\bar{y}_2^t & + \\
bx_1^t|t & + & bx_1^t|c & + & bx_1^t|dy_1^t & + & \delta & + & bx_1^t|\bar{y}_2^t & + \\
\delta & + & \delta & + & \delta & + & \delta & + & \delta & + \\
\bar{x}_2^t|t & + & \bar{x}_2^t|c & + & \bar{x}_2^t|dy_1^t & + & \delta & + & \bar{x}_2^t|\bar{y}_2^t & .
\end{array}$$

To see that $\tau_{\{t\}}(\bar{x}^t|\bar{y}^t) = \bar{x}|\bar{y}$ we can inspect the summands separately (since $\tau_{\{t\}}$ distributes over +). Indeed, $a|\tau = \tau_{\{t\}}(a|t) = \delta$; and e.g. $\bar{x}_2|dy_1 = \tau_{\{t\}}(\bar{x}_2^t|dy_1^t)$ follows by the induction hypothesis, using the fact that

$dy_1^t = \overline{dy}_1^t$.

In the same way one can prove the following proposition which generalises Proposition 3.6(i) and is of independent interest:

3.7. PROPOSITION. *Let* $I \subseteq A$ *be such that* $I|A = \{\delta\}$. (*Here* $I|A = \{c \mid \exists i \in I, a \in A\ i|a = c\}$.) *Then in* A_τ^ω:

$$\tau_I(x\|y) = \tau_I(\tau_I(x)\|\tau_I(y)).$$

(ii) *Moreover, let* $(A|A) \cap I = \emptyset$. *Then in* A_τ^ω:

$$\tau_I(x\|y) = \tau_I(x)\|\tau_I(y).$$

3.8. COROLLARY. $A_\tau^\omega \models x\|(y\|z) = (x\|y)\|z$.

PROOF. Let t be as in Proposition 3.6. Note that Proposition 3.6(i) entails $(x\|y)^t = x^t\|y^t$. Now:

$$x\|(y\|z) = \tau_{\{t\}}(x^t\|(y\|z)^t) = \tau_{\{t\}}(x^t\|(y^t\|z^t)) \underset{(*)}{=}$$

$$\tau_{\{t\}}((x^t\|y^t)\|z^t) = (x\|y)\|z.$$

Here (*) follows from the associativity of $\|$ in ACP (see [2]).□

3.9. EXPANSION THEOREM FOR ACP$_\tau$. *Let communication be binary. Then in* A_τ^ω:

$$x_1\|\ldots\|x_k = \sum_{1\leq i\leq k} x_i ⌊⌊ X_k^i + \sum_{1\leq i<j\leq k} (x_i|x_j) ⌊⌊ X_k^{i,j}$$

where X_k^i *is the merge of* $x_1,\ldots,x_k$ *except* x_i *and* $X_k^{i,j}$ *is the merge of* $x_1,\ldots,x_k$ *except* x_i, x_j $(k \geq 3)$.

PROOF. $x_1\|\ldots\|x_k = \overline{x}_1\|\ldots\|\overline{x}_k = \tau_{\{t\}}(\overline{x}_1^t\|\ldots\|\overline{x}_k^t) \underset{(*)}{=}$

$$\tau_{\{t\}}(\sum \overline{x}_i^t ⌊⌊ (\overline{X}_k^i)^t + \sum (\overline{x}_i^t|\overline{x}_j^t) ⌊⌊ (\overline{X}_k^{i,j})^t) =$$

$$\sum \tau_{\{t\}}(\overline{x}_i^t ⌊⌊ (\overline{X}_k^i)^t) + \sum \tau_{\{t\}}((\overline{x}_i^t|\overline{x}_j^t) ⌊⌊ (\overline{X}_k^{i,j})^t) \underset{(**)}{=}$$

$$\sum(\tau_{\{t\}}(\overline{x}_i^t) ⌊⌊ \tau_{\{t\}}(\overline{X}_k^i)^t) + \sum(\tau_{\{t\}}\overline{x}_i^t | \tau_{\{t\}}\overline{x}_j^t) ⌊⌊ \tau_{\{t\}}(\overline{X}_k^{i,j})^t$$

$$\sum \bar{x}_i \mathbin{\|\!\!\_} \bar{x}_k^i \quad + \quad \sum (\bar{x}_i | \bar{x}_j) \mathbin{\|\!\!\_} \bar{x}_k^{i,j} =$$

$$\sum x_i \mathbin{\|\!\!\_} x_k^i \quad + \quad \sum (x_i | x_j) \mathbin{\|\!\!\_} x_k^{i,j}.$$

Here (*) is the Expansion Theorem for ACP (see [5]) and (**) is by Proposition 3.6. □

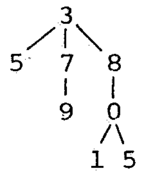
APPENDIX I. TERMINATION OF ACP_τ REDUCTIONS PROVED BY RECURSIVE PATH ORDERINGS

In this Appendix we will prove the termination result in the Elimination Theorem 2.20 by the method of recursive path orderings as in DERSHOWITZ [7]. Since we will give a slightly different presentation of recursive path orderings, a short account of this method will be given. Our presentation replaces Dershowitz's inductive definition of the recursive path ordering by a reduction procedure (which may be seen as an 'operationalisation' of that inductive definition). This reduction procedure provides a somewhat easier notation in applications.

We start with the basis of the recursive path ordering method, the Kruskal Tree Theorem. First we need a definition:

1. DEFINITION. (i) Let D be the domain of finite commutative rooted trees whose nodes are labeled with natural numbers; alternatively one may consider an element t of D as a partially ordered multiset of natural numbers such that t has a least element.

Example: t =



We will use the self-explaining notation $t = 3(5,7(9),8(0(1,5)))$. This notation is ambiguous since the 'arguments' of the 'operators' may be permuted, e.g. also $t = 3(8(0(5,1)),5,7(9))$.

(ii) Let $t,s \in D$. We say that s is covered by t, notation $s \sqsubseteq t$, if there is an injection φ: NODES(s) $\longrightarrow$ NODES(t) which is an order-preserving isomorphism and such that for all nodes $\alpha \in$ NODES(s) we have: label(α) $\geqslant$ label($\varphi(\alpha)$) where $\geqslant$ is the ordering on $\mathbb{N}$.

Example: $s = 2(9,7(4,0)) \sqsubseteq t$ as in (i):

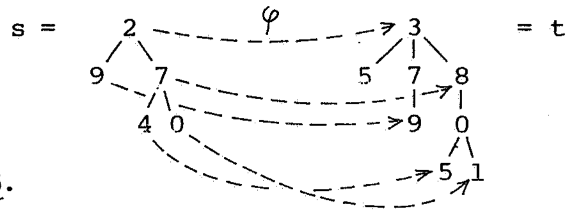


Figure 15.

(Note that the embedding φ is unique in this case.)

Clearly, $\sqsubseteq$ is a p.o. on D. Now there is the following beautiful theorem:

2. <u>KRUSKAL TREE THEOREM</u>. <u>Let</u> $t_1, t_2, t_3, \ldots$ <u>be a sequence in D. Then for some</u> $i < j$: $t_i \sqsubseteq t_j$.

In fact, this is not the most general formulation of the theorem; see DERSHOWITZ [7]. The formulation there is stronger in two respects: the linear ordering of the labels (in our case $\mathbb{N}$) can be taken to be a partial order which is well-founded; and secondly, Kruskal's original formulation concerns noncommutative trees and an embedding φ as above must also respect the 'left-to-right' ordering. Clearly, that version implies immediately the above statement of the Tree Theorem. For a short proof see DERSHOWITZ [8].

The next definition is from [7]:

3. <u>DEFINITION</u>. The p.o. $\triangleright$ on D is defined inductively as follows:
$t = n(t_1, \ldots, t_k) \triangleright m(s_1, \ldots s_\ell) = s$ $(k, \ell \geqslant 0)$ iff

(i) $n > m$ and $t \triangleright s_i$ for all $i = 1, \ldots, \ell$

or

(ii) $n = m$ and $\{t_1, \ldots, t_k\} \triangleright\!\triangleright \{s_1, \ldots, s_\ell\}$ where $\triangleright\!\triangleright$ is the p.o. on multisets of elements of D induced by $\triangleright$,

or

(iii) $n < m$ and $t_i \trianglerighteq s$ for some $i \in \{1, \ldots, k\}$.

It is implicit in [7] that an equivalent definition of $\triangleright$ is:

4. <u>DEFINITION</u>. The p.o. $\triangleright$ on D is defined inductively as follows:
(a) $t = n(t_1, \ldots, t_k) \triangleright m(s_1, \ldots, s_\ell) = s$ $(k, \ell \geqslant 0)$ iff

(i) as above

or

(ii) as above

or

(iii)' $s = t_i$ for some $i \in \{1, \ldots, k\}$.

(b) $\triangleright$ is transitive.

(Here the cases (i), (ii), (iii)' may overlap. The transitivity has to be required explicitly now.)

5. <u>EXAMPLE</u>. $t = 5(6,7(8)) \rhd 4(6,5(6,8),4(6,6(8,8,8,8))) = s$

<u>Proof</u>: By (i) from Definition 3, $t \rhd s$ if: (a) $t \rhd 6$ and (b) $t \rhd 5(6,8)$ and (c) $t \rhd 4(6,6(8,8,8,8))$.

(a) follows by (iii) of Definition 3; (b) follows by (ii) and $7(8) \rhd 8$ (by (iii)).

(c) follows from (d) $t \rhd 6$ and (e) $t \rhd 6(8,8,8,8)$.

(d) is by (iii) and (e) is so by (iii) since $7(8) \rhd 6(8,8,8,8)$ (by (i),(iii)).

So, establishing that $t \rhd s$ requires a miniature proof. Another presentation may be more convenient: instead of by the inductive definition above we can also define $\rhd$ by an auxiliary reduction procedure as follows.

Let D^* be D where some nodes of $t \in D$ may be marked with $*$. E.g.

$3^*(1,2^*(4)) = 3^*(1,2^*(4)) \in D^*$.

<u>Notation</u>: if $t = n(t_1,\dots,t_k)$ or $t = n^*(t_1,\dots,t_k)$, then $t^* = n^*(t_1,\dots,t_k)$.

(The marker $*$ can be understood as a command to replace the marked term by a lesser term.)

6. <u>DEFINITION</u>. On D^* a reduction relation $\Longrightarrow$ is defined as follows.

(0) $n(t_1,\dots,t_k) \Longrightarrow n^*(t_1,\dots,t_k)$ $(k \geqslant 0)$

(1) if $n > m$ then $n^*(t_1,\dots,t_k) \Longrightarrow m(n^*(\vec{t}),\dots,n^*(\vec{t}))$
$(k \geqslant 0,\ s \geqslant 0$ copies of $n^*(\vec{t}))$

(2) $n^*(t_1,\dots,t_k) \Longrightarrow n(t_1^*,\dots,t_1^*,t_2,\dots,t_k)$ $(k \geqslant 1,\ s \geqslant 0$ copies of $t_1^*)$

(3) $n^*(t_1,\dots,t_k) \Longrightarrow t_i$ $(i \in \{1,\dots,k\},\ k \geqslant 1)$

(4) if $t \Longrightarrow s$ then $n(--,t,--) \Longrightarrow n(--,s,--)$.

Furthermore, $\Longrightarrow\!\!\!\!\!\Rightarrow$ is the transitive reflexive closure of $\Longrightarrow$.

(In fact, (4) is superfluous for the definition of $\Rightarrow\!\!\!\gg$; without it one easily derives: if $t \Rightarrow\!\!\!\gg s$ then $n(--,t,--) \Rightarrow\!\!\!\gg n(--,s,--)$.)

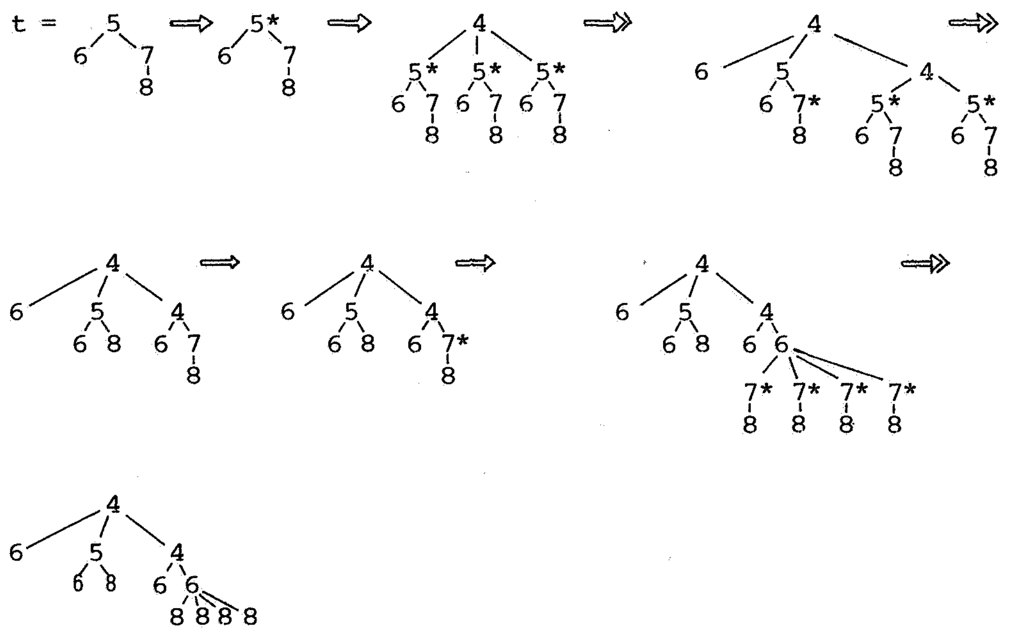
We are only interested in *-free $t \in D \subseteq D^*$. Now we have by a tedious but routine proof which is omitted:

7. PROPOSITION. Let $t,s \in D$ (i.e. not containing *). Then:

$$t \Rightarrow\!\!\!\gg s \text{ iff } t \rhd s.$$

8. EXAMPLE. (i) $4 \Rightarrow 4^* \Rightarrow 3(4^*,4^*) \Rightarrow 3(2(4^*),4^*) \Rightarrow 3(2(1),4^*) \Rightarrow 3(2(1),0)$.

(ii) Cf. Example 5:



In DERSHOWITZ [7] the following facts about $\rhd$ are proved:

9. PROPOSITION. $\rhd$ is a partial order.

The proof requires a simple induction to show the irreflexivity.

10. PROPOSITION. (i) $n(t_1,\ldots,t_k) \rhd n(t_2,\ldots,t_k)$

(ii) $n(t_1,\ldots,t_k) \rhd t_i \quad (1 \leqslant i \leqslant k)$

(iii) $t > s \Rightarrow n(\ldots,t,\ldots) \rhd n(\ldots,s,\ldots)$

(iv) if $n > m$ then $n(t_1,\ldots,t_k) \rhd m(t_1,\ldots,t_k)$.

PROOF. Using Proposition 7, (i)-(iii) are immediate; e.g. (ii): $n(\vec{t}) \Longrightarrow n^*(\vec{t}) \Longrightarrow t_i$ and (i): $n(t_1,\ldots,t_k) \Longrightarrow n^*(t_1,\ldots,t_k) \Longrightarrow n(t_2,\ldots,t_k)$. As to (iv): $n(\vec{t}) \Longrightarrow n^*(\vec{t}) \Longrightarrow m(n^*(\vec{t}),\ldots,n^*(\vec{t})) \Longrightarrow m(t_1,\ldots,t_k)$. □

Using Proposition 10 one shows easily:

11. PROPOSITION. $s \sqsubseteq t \implies t \trianglerighteq s$.

From this we have

12. THEOREM (Dershowitz). (The termination property for the recursive path ordering $\triangleright$) $\triangleright$ is a well-founded partial order.

PROOF. Suppose $t_0 \triangleright t_1 \triangleright t_2 \triangleright\ldots$ is an infinite descending chain w.r.t. $\triangleright$. Then, by the Kruskal Tree Theorem 2, $t_i \sqsubseteq t_j$ for some $i<j$. So by Proposition 11, $t_j \trianglerighteq t_i$. However since $\triangleright$ is a p.o., this contradicts $t_i > t_j$. □

13. Application to ACP_τ. We want to prove that the rewrite rules (from left to right) associated to the axioms of ACP_τ except A1,2,5, C1,2 and T1,2,3 are terminating. These rewrite rules have, in tree notation, the following form: (see Table 4, next page).

Note that the occurrence of $\|$ in the RHS of the rules CM3, CM7 prevents us to order the operators directly in a way suitable for an application of the termination property of recursive path orderings. Instead, we have to rank the operators $\|$, $\lfloor\!\lfloor$, $|$ simply by (e.g.) the natural number that is the sum $|x|+|y|$ of the lengths of the arguments x,y. Here $|x|$ is inductively defined by:

$$|a| = |\tau| = 1$$

$$|x \square y| = |x| + |y| \text{ for } \square = +,\cdot,\|,\lfloor\!\lfloor,|$$

$$|\partial_H(x)| = |\tau_I(x)| = |x|.$$

The ranked operators $\|_n, \lfloor\!\lfloor_n, |_n, +, \cdot, \partial_H, \tau_I$ are partially ordered as follows:

$$\|_n > \lfloor\!\lfloor_n, |_n$$

$$\lfloor\!\lfloor_n, |_n > \|_{n-1}$$

$$\|_n, \lfloor\!\lfloor_n, |_n > \cdot > +$$

$$\partial_H, \tau_I > \cdot\,.$$

(See Figure 16.)

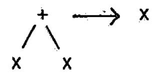
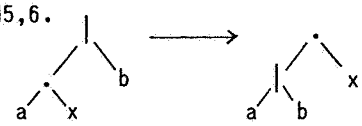
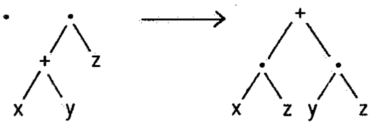
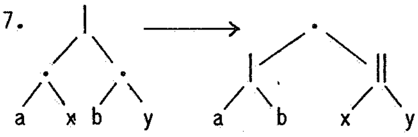
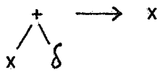
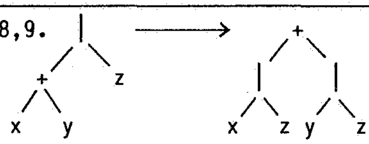
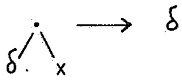
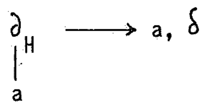
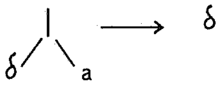
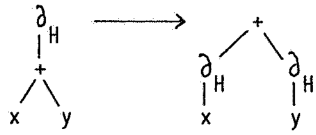
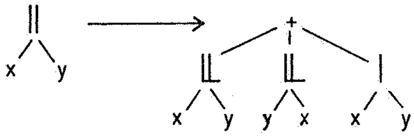
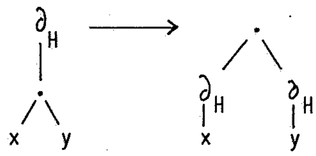
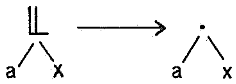
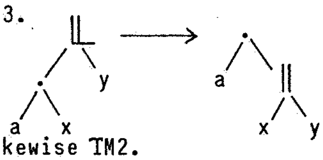
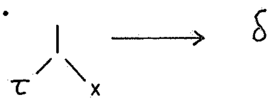
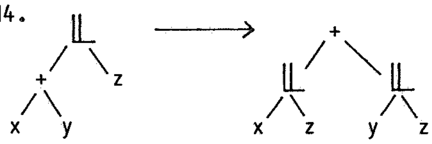
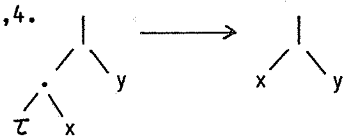
A3. 	CM5,6. 
A4. 	CM7. 
A6. 	CM8,9. 
A7. 	D1,2.  Likewise DT.
C3. 	D3. 
CM1. 	D4. 
CM2.  Likewise TM1.	TI1-5: analogous to DT, D1-4.
CM3.  Likewise TM2.	TC1,2. 
CM4. 	TC3,4. 

Table 4: Rewrite rules associated to the axioms of $ACP_\tau - \{A1,2,5;\ C1,2;\ T1,2,3\}$.

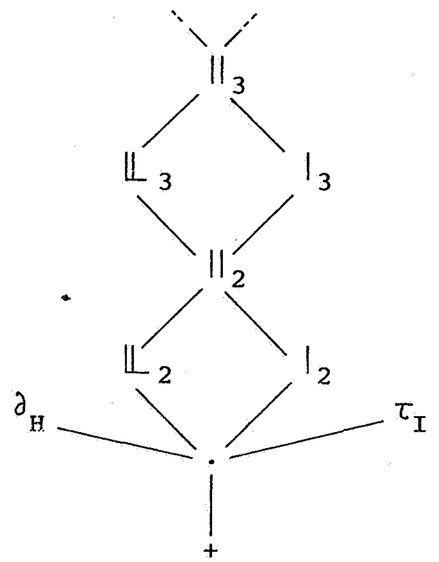


Figure 16.

Now consider a closed ACP_τ-term T. Rank all $\parallel, \lfloor\!\lfloor, |$-operators in T by the sum of the norms $|.|$ of their arguments.

Example: $T = (ab \lfloor\!\lfloor cd) \lfloor\!\lfloor (\tau q | (r + uv))$ will be ranked as

$$T_r = (ab \lfloor\!\lfloor_4 cd) \lfloor\!\lfloor_9 (\tau q |_5 (r + uv)).$$

To T_r we associate an element $t \in D$ by writing down the formation tree of T_r:

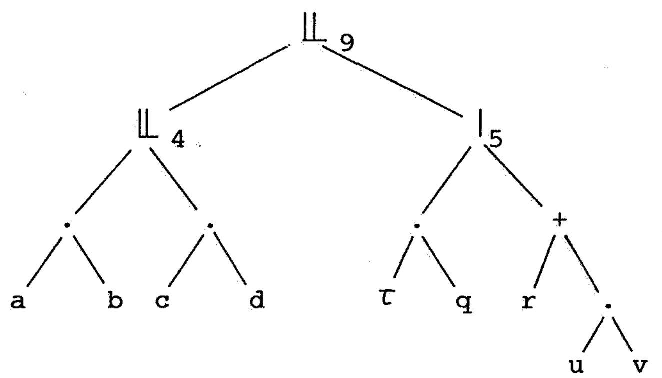


Figure 17.

(In fact, we must assign to the $a, \tau, \parallel_n, \lfloor\!\lfloor_n, |_n, +, \cdot, \partial_H, \tau_I$ natural numbers corresponding to the p.o. in Figure 16 above. To all atoms we assign, say, 0.)

Now we have:

13.1. THEOREM. The rewrite rules in Table 4 have the termination property.

PROOF. Let $\triangleright$ be the recursive path ordering induced by the p.o. on the ranked operators as defined above. We will show that for each closed instance $t \dashrightarrow s$ of the rewrite rules, we have $t \triangleright s$. In order to do so, we use the alternative definition of $\triangleright$ as $\overset{+}{\Longrightarrow}$ (the transitive closure of $\Longrightarrow$). We will treat some typical cases.

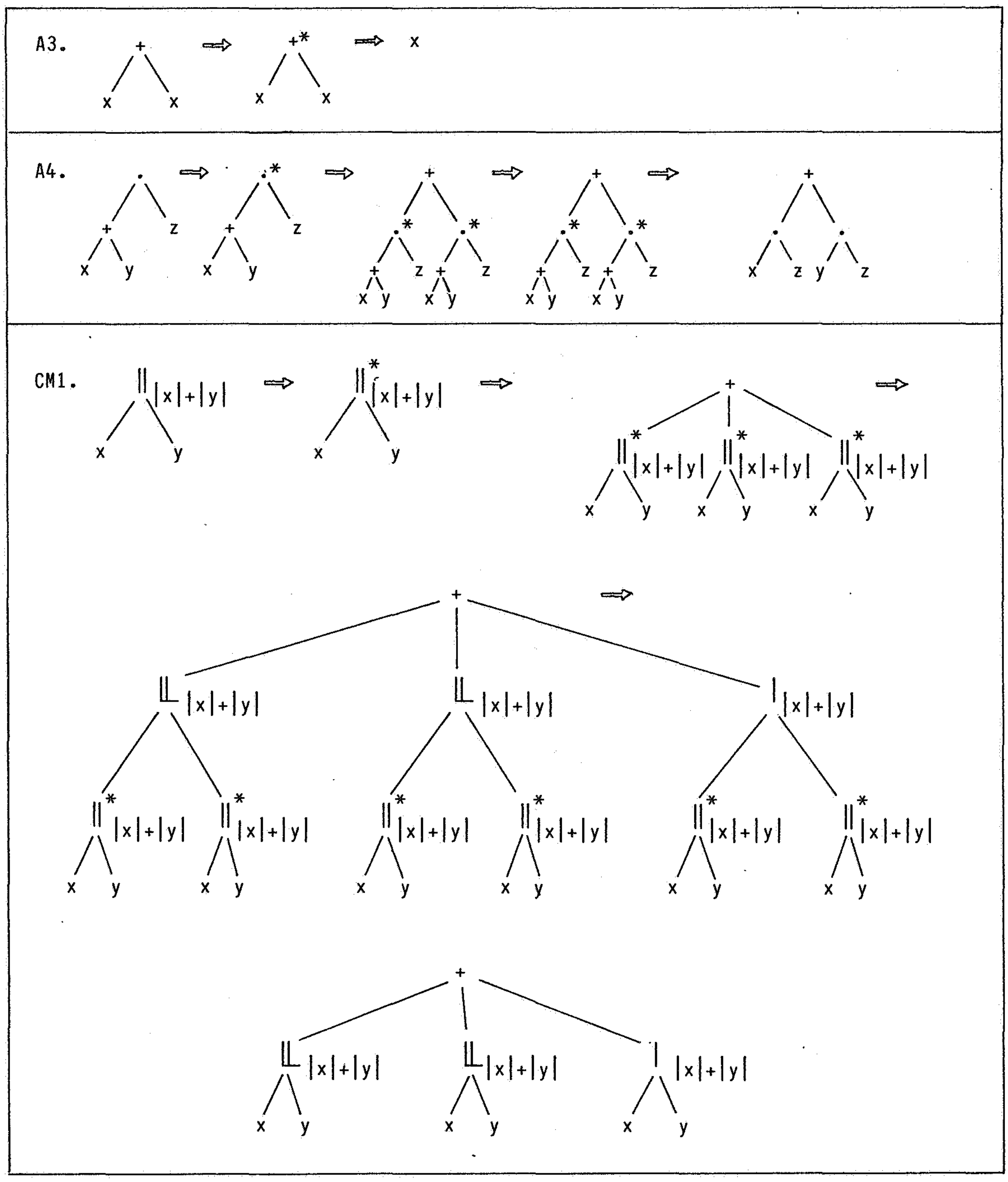


Table 5.

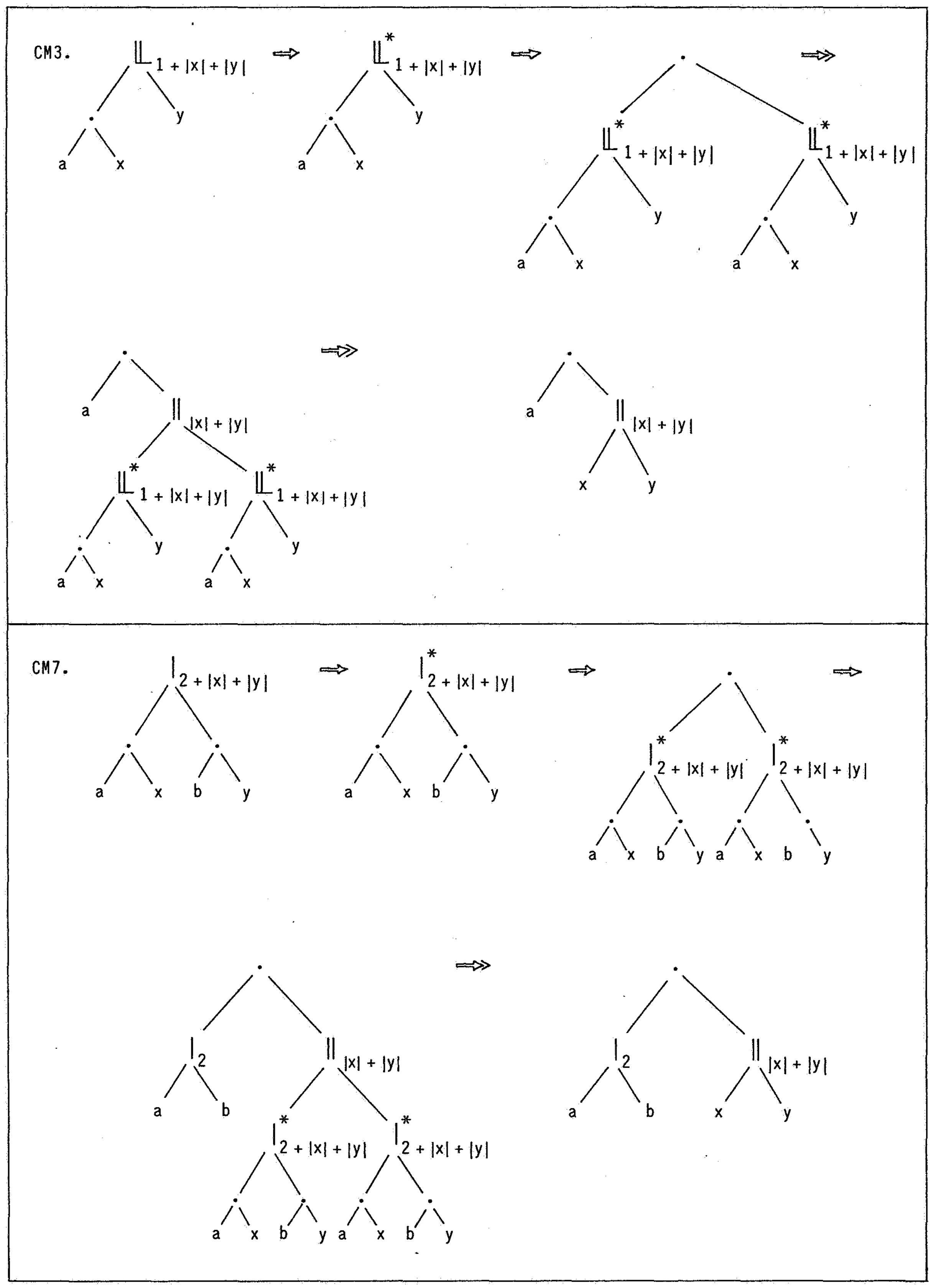


Table 6.

APPENDIX II. AN INDUCTIVE PROOF OF ASSOCIATIVITY OF MERGE IN ACP_τ

We will prove that in ACP_τ the following identities between closed terms are derivable:

(1)	$(x \lfloor\lfloor y) \lfloor\lfloor z = x \lfloor\lfloor (y \| z)$
(2)	$(x \mid ay) \lfloor\lfloor z = x \mid (ay \lfloor\lfloor z)$
(3)	$x \mid y = y \mid x$
(4)	$x \| y = y \| x$
(5)	$x \mid (y \mid z) = (x \mid y) \mid z$
(6)	$x \| (y \| z) = (x \| y) \| z$

Table 7.

These are the axioms of standard concurrency as in Table 3 (Section 3), except for (2) which is a special case of the second axiom of standard concurrency. (Alternatively, (2) may be replaced by:

$$(x \mid y) \lfloor\lfloor z = x \mid (y \lfloor\lfloor z) \text{ if } y \text{ is stable.}$$

Here y is 'stable', in the terminology of MILNER [12], if it does not start with a τ-step.)

In Corollary 3.8 a different proof of (6) is given. The present proof uses an essentially straightforward induction to the lengths of the terms involved; the induction has to be simultaneously applied to several of (1)-(6). These identities, however, are interesting in their own right.

The proof has two main parts; in the first and easiest part, identities (3),(4),(5) are proved. The second part takes care of the main identity, (6); the proof is complicated by the fact that we have in ACP_τ only the weak version (2) of the second axiom of standard concurrency.

All identities (1)-(6) are proved for basic terms $\in T$ (see Definition 3.1). In view of the Elimination Theorem 2.20 this entails the identities for all closed ACP_τ-terms x,y,z.

1. <u>PROPOSITION. Let $x,y,z \in T$. Then:</u>

(i) $ACP_\tau \vdash x \mid y = y \mid x$

(ii) $ACP_\tau \vdash x \| y = y \| x$.

PROOF. Let $|x|$ be the length in symbols of x (see Definition in Appendix I,13). The proof uses an induction to $|x| + |y|$. We prove (i),(ii) simultaneously.

The induction hypothesis is: (i),(ii) are proved for all x',y' such that $|x'| + |y'| < |x| + |y|$. First we will prove the induction step of (i), $x|y = y|x$.

Case 1. $x = x_1 + x_2$. So $|x_i| < |x|$, $i = 1,2$. Then $x|y = (x_1 + x_2)|y = x_1|y + x_2|y$

$= \text{(ind. hyp.)}\ y|x_1 + y|x_2 = y|(x_1 + x_2) = y|x$.

Case 2. $y = y_1 + y_2$: similar.

Case 3. $x = \tau$: $x|y = \tau|y = \delta = y|\tau = y|x$.

Case 4. $y = \tau$: similar.

Case 5. $x = \tau x'$: $x|y = \tau x'|y = x'|y = y|x' = y|\tau x' = y|x$.

Case 6. $x = a$, $y = b$: $x|y = a|b = b|a = y|x$.

Case 7. $x = ax'$, $y = by'$: $x|y = ax'|by' = (a|b)(x'\|y') = (b|a)(y'\|x') = y|x$.

Case 8. $x = a$, $y = by'$: $x|y = (a|b)y' = (b|a)y' = y|x$.

Case 9. $x = ax'$, $y = b$: similar.

(Note that in case 7 the induction hypothesis for (ii) is used.)

Next to show (ii) $x\|y = y\|x$:

$x\|y = x \mathbin{\|\!\_} y + y \mathbin{\|\!\_} x + x|y = y \mathbin{\|\!\_} x + x \mathbin{\|\!\_} y + y|x = y\|x$. $\square$

2. PROPOSITION. Let $x,y,z \in T$. Then $ACP_\tau \vdash x|(y|z) = (x|y)|z$.

PROOF. Induction on $|x| + |y| + |z|$.

Case 1. $x = x_1 + x_2$. Then $x|(y|z) = x_1|(y|z) + x_2|(y|z) = (x_1|y)|z + (x_2|y)|z =$ $((x_1|y) + (x_2|y))|z = ((x_1 + x_2)|y)|z = (x|y)|z$.

Case 2. Similar with y and z sums of smaller terms.

Case 3. x,y,z have one of the forms $a, \tau, au, \tau u$. We mention one of the 4^3 cases: $(\tau x'|ay')|b = (x'|ay')|b = x'|(ay'|b) = \tau x'|(ay'|b)$. Note that one of the cases is just axiom C2 from ACP_τ (Table 2). $\square$

For the second half of the proof we need two preparatory propositions.

3. DEFINITION. Let x,y be closed ACP_τ-terms. Then we define: $ACP_\tau \vdash x \sqsubseteq y$ if for some closed term z, $ACP_\tau \vdash y = x + z$.

3.1. REMARK. Note the difference with $\sqsubseteq$ as defined for T, in Definition 3.2. The present 'summand inclusion', $ACP_\tau \vdash .. \sqsubseteq ..$, is just $\sqsubseteq$ modulo ACP_τ-equality. In the sequel we will sometimes write $x \sqsubseteq y$ where $ACP_\tau \vdash x \sqsubseteq y$ is meant, if it is clear that we are working modulo ACP_τ-equality.

4. EXAMPLE. (i) $ACP_\tau \vdash a \sqsubseteq \tau a$ (since $a = a + \tau a$)

(ii) $ACP_\tau \vdash a \sqsubseteq a \| \tau$ (since $a\|\tau = \tau a + a\tau + a|\tau = \tau a + a$)

(iii) $ACP_\tau \vdash \delta \sqsubseteq x$, for all x

(iv) $ACP_\tau \vdash a + \tau a + \tau b \mathrel{\substack{\sqsubseteq\\\sqsupseteq}} b + \tau a + \tau b$.

5. PROPOSITION. Let x,y be closed terms. Then:

$$ACP_\tau \vdash x \sqsubseteq y \;\;\&\;\; ACP_\tau \vdash y \sqsubseteq x \;\Rightarrow\; ACP_\tau \vdash x = y.$$

PROOF. We may suppose, by the Elimination Theorem 2.20, that $x,y \in T$. Suppose $ACP_\tau \vdash y = x + z$ for some $z \in T$ and $ACP_\tau \vdash x = y + u$ for some $u \in T$. Then $ACP_\tau \vdash x = x + z + u$. Therefore the process trees corresponding to x and $x + z + u$ bisimulate: $[x] \leftrightarrows_{r,\tau} [x + z + u]$. (Here $[x]$ is the interpretation of x in the graph domain $\mathcal{G}$ as in Section 2; since $x \in T$ this is a process tree.) Say R is a r,τ-bisimulation between $[x]$ and $[x + z + u] = [x] + [z] + [u]$. Let R' be the restriction of R to (the node sets of) $[x]$ and $[x] + [z]$. Now R' need not be a bisimulation between these trees; however if I is the trivial (identity) bisimulation between $[x]$ with itself, then it is not hard to see that $R' \cup I$ is a r,τ-bisimulation between $[x]$ and $[x] + [z] = [x + z]$. (Alternatively: let R be a bisimulation as indicated which is maximal w.r.t. inclusion. Then the restriction R' is a bisimulation as desired.)

Hence $ACP_\tau \vdash x = x + z = y$. □

6. PROPOSITION. Let x be a closed term. Then $ACP_\tau \vdash x \mathbin{⌊\!⌊} \tau = x$.

PROOF. We may suppose $x \in T$, and use induction on $|x|$.
If $x = x_1 + x_2$ then $x \mathbin{⌊\!⌊} \tau = x_1 \mathbin{⌊\!⌊} \tau + x_2 \mathbin{⌊\!⌊} \tau = x_1 + x_2 = x$.
If $x = a$ then $a \mathbin{⌊\!⌊} \tau = a\tau = a$.
If $x = ax'$ then $ax' \mathbin{⌊\!⌊} \tau = a(x' \| \tau) = a(x' \mathbin{⌊\!⌊} \tau + \tau \mathbin{⌊\!⌊} x' + x'|\tau) = a(x' \mathbin{⌊\!⌊} \tau + \tau x' + \delta) = a(x' + \tau x') = a\tau x' = ax'$.
The cases $x = \tau$, $x = \tau x'$ are similar. □

We will now start the simultaneous proof of (1),(2),(6) in Table 7.

7. THEOREM. Let x,y,z be closed ACP_τ-terms and $a \in A$. Then:

(i) $ACP_\tau \vdash (x \mathbin{\lfloor\!\lfloor} y) \| z = x \mathbin{\lfloor\!\lfloor} (y \| z)$

(ii) $ACP_\tau \vdash (x | ay) \mathbin{\lfloor\!\lfloor} z = x | (ay \mathbin{\lfloor\!\lfloor} z)$

(iii) $ACP_\tau \vdash x \| (y \| z) = (x \| y) \| z$.

PROOF. We may assume $x,y,z \in T$; this makes an induction to $|x| + |y| + |z|$ possible. We will prove (i)-(iii) by a simultaneous induction. Let the induction hypothesis be that (i)-(iii) are proved for all $x',y',z' \in T$ such that $|x'| + |y'| + |z'| < |x| + |y| + |z|$.

First we prove the induction step (i): $(x \mathbin{\lfloor\!\lfloor} y) \| z = x \mathbin{\lfloor\!\lfloor} (y \| z)$.

Case (i)1. $x = x_1 + x_2$. Then $(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z = (x_1 \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z + (x_2 \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z =$ (ind. hyp.) $x_1 \mathbin{\lfloor\!\lfloor} (y \| z) + x_2 \mathbin{\lfloor\!\lfloor} (y \| z) = (x_1 + x_2) \mathbin{\lfloor\!\lfloor} (y \| z)$.

Case (i)2. $x = \tau$. Then: $(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z = \tau y \mathbin{\lfloor\!\lfloor} z = \tau(y \| z) = \tau \mathbin{\lfloor\!\lfloor} (y \| z) = x \mathbin{\lfloor\!\lfloor} (y \| z)$.

Case (i)3. $x = \tau x'$. Then: $(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z = \tau(x' \| y) \mathbin{\lfloor\!\lfloor} z = \tau((x' \| y) \| z) = \tau(x' \| (y \| z)) = \tau x' \mathbin{\lfloor\!\lfloor} (y \| z) = x \mathbin{\lfloor\!\lfloor} (y \| z)$.

The cases $x = a$, $x = ax'$ are similar. This ends the proof of the induction step (i).

Next consider the induction step (ii): $(x | ay) \mathbin{\lfloor\!\lfloor} z = x | (ay \mathbin{\lfloor\!\lfloor} z)$.
This will again be proved by a case distinction according to the formation of $x \in T$: $x = x_1 + x_2$, $x = \tau$, $\tau x'$, b, or bx'.

Case (ii)1. $x = x_1 + x_2$. Then $x | (ay \mathbin{\lfloor\!\lfloor} z) = (x_1 + x_2) | (ay \mathbin{\lfloor\!\lfloor} z) = x_1 | (ay \mathbin{\lfloor\!\lfloor} z) + x_2 | (ay \mathbin{\lfloor\!\lfloor} z) = (x_1 | ay) \mathbin{\lfloor\!\lfloor} z + (x_2 | ay) \mathbin{\lfloor\!\lfloor} z = (x_1 | ay + x_2 | ay) \mathbin{\lfloor\!\lfloor} z = ((x_1 + x_2) | ay) \mathbin{\lfloor\!\lfloor} z = (x | ay) \mathbin{\lfloor\!\lfloor} z$.

Case (ii)2. $x = \tau$. Then $(x | ay) \mathbin{\lfloor\!\lfloor} z = x | (ay \mathbin{\lfloor\!\lfloor} z) = \delta$.

Case (ii)3. $x = \tau x'$. Then $(x | ay) \mathbin{\lfloor\!\lfloor} z = (\tau x' | ay) \mathbin{\lfloor\!\lfloor} z = (x' | ay) \mathbin{\lfloor\!\lfloor} z = x' | (ay \mathbin{\lfloor\!\lfloor} z) = \tau x' | (ay \mathbin{\lfloor\!\lfloor} z) = x | (ay \mathbin{\lfloor\!\lfloor} z)$.

Case (ii)4. $x = b$. Then $(x | ay) \mathbin{\lfloor\!\lfloor} z = (b | ay) \mathbin{\lfloor\!\lfloor} z = (b | a)y \mathbin{\lfloor\!\lfloor} z = (b | a)(y \| z)$, and also $x | (ay \mathbin{\lfloor\!\lfloor} z) = b | (ay \mathbin{\lfloor\!\lfloor} z) = b | (a(y \| z)) = (b | a)(y \| z)$.

Case (ii)5. $x = bx'$. Then $(x | ay) \mathbin{\lfloor\!\lfloor} z = (bx' | ay) \mathbin{\lfloor\!\lfloor} z = (b | a)(x' \| y) \mathbin{\lfloor\!\lfloor} z = (b | a)((x' \| y) \| z)$, and $x | (ay \mathbin{\lfloor\!\lfloor} z) = bx' | (ay \mathbin{\lfloor\!\lfloor} z) = bx' | a(y \| z) = (b | a)(x' \| (y \| z))$. By the induction hypothesis for statement (iii) therefore $(x | ay) \mathbin{\lfloor\!\lfloor} z = x | (ay \mathbin{\lfloor\!\lfloor} z)$.

This ends the proof of the induction step (ii).

Now consider the induction step (iii): $L = x\|(y\|z) = (x\|y)\|z = R$. By the axioms in ACP_τ, we have:

$$L = x\|(y\|z) = x \mathbin{|\!\!\lfloor} (y\|z) + (y\|z) \mathbin{|\!\!\lfloor} x + x|(y\|z) =$$
$$x \mathbin{|\!\!\lfloor} (y\|z) + (y \mathbin{|\!\!\lfloor} z + y|z + z \mathbin{|\!\!\lfloor} y) \mathbin{|\!\!\lfloor} x + x|(y \mathbin{|\!\!\lfloor} z + z \mathbin{|\!\!\lfloor} y + y|z) =$$
$$x \mathbin{|\!\!\lfloor} (y\|z) + (y \mathbin{|\!\!\lfloor} z) \mathbin{|\!\!\lfloor} x + (y|z) \mathbin{|\!\!\lfloor} x + (z \mathbin{|\!\!\lfloor} y) \mathbin{|\!\!\lfloor} x + x|(y \mathbin{|\!\!\lfloor} z) +$$
$$x|(z \mathbin{|\!\!\lfloor} y) + x|(y|z).$$

Likewise R can be expanded. We will use the following abbreviations: $L = \ell_1 + \ldots + \ell_7$ and $R = r_1 + \ldots + r_7$ where

$$\begin{array}{llll}
\ell_1 = & x \mathbin{|\!\!\lfloor} (y\|z) & r_1 = & (x \mathbin{|\!\!\lfloor} y) \mathbin{|\!\!\lfloor} z \\
\ell_2 = & (y \mathbin{|\!\!\lfloor} z) \mathbin{|\!\!\lfloor} x & r_2 = & (x|y) \mathbin{|\!\!\lfloor} z \\
\ell_3 = & (y|z) \mathbin{|\!\!\lfloor} x & r_3 = & (y \mathbin{|\!\!\lfloor} x) \mathbin{|\!\!\lfloor} z \\
\ell_4 = & (z \mathbin{|\!\!\lfloor} y) \mathbin{|\!\!\lfloor} x & r_4 = & z \mathbin{|\!\!\lfloor} (x\|y) \\
\ell_5 = & x|(y \mathbin{|\!\!\lfloor} z) & r_5 = & (x \mathbin{|\!\!\lfloor} y)|z \\
\ell_6 = & x|(z \mathbin{|\!\!\lfloor} y) & r_6 = & (y \mathbin{|\!\!\lfloor} x)|z \\
\ell_7 = & x|(y|z) & r_7 = & (x|y)|z
\end{array}$$

Claim. $\ell_i \sqsubseteq R$, for $i = 1,\ldots,7$.

From the claim the induction step (iii) follows at once. Namely, we then have: $x\|(y\|z) \sqsubseteq (x\|y)\|z$, hence by Proposition 1(ii): $x\|(y\|z) \sqsubseteq z\|(x\|y)$ (*). Now $z\|(x\|y) = z\|(y\|x) \sqsubseteq x\|(z\|y) = x\|(y\|z)$, where '$\sqsubseteq$' follows from (*). So we have $x\|(y\|z) \equiv (x\|y)\|z$, and by Proposition 5: $x\|(y\|z) = (x\|y)\|z$.

The remainder of the proof is devoted to:

Proof of the claim.

(a) $\ell_7 = r_7 \sqsubseteq R$ by Proposition 2.

(b) $\ell_1 = r_1 \sqsubseteq R$ is statement (i) of this theorem; this induction step has already been proved. Likewise for $\ell_2 = r_3 \sqsubseteq R$ and $\ell_4 = r_4 \sqsubseteq R$.

(c) $\ell_3 \sqsubseteq r_6 \sqsubseteq R$. Here $\ell_3 = (z|y) \mathbin{|\!\!\lfloor} x$ and $r_6 = z|(y \mathbin{|\!\!\lfloor} x)$.
Induction on z:

<u>Case (iii)(c)1.</u> $z = z_1 + z_2$. Then $\ell_3 = ((z_1+z_2)|y) \lfloor\!\lfloor x = (z_1|y) \lfloor\!\lfloor x +$
$(z_2|y) \lfloor\!\lfloor x \sqsubseteq$ (ind. hyp.) $z_1|(y \lfloor\!\lfloor x) + z_2|(y \lfloor\!\lfloor x) =$
$(z_1+z_2)|(y \lfloor\!\lfloor x) = z|(y \lfloor\!\lfloor x)$.

<u>Case (iii)(c)2.</u> $z = \tau$. Then $\ell_3 = r_6 = \delta$.

<u>Case (iii)(c)3.</u> $z = \tau z'$. Then $\ell_3 = (\tau z'|y) \lfloor\!\lfloor x = (z'|y) \lfloor\!\lfloor x \sqsubseteq z'|(y \lfloor\!\lfloor x) =$
$\tau z'|(y \lfloor\!\lfloor x) = z\ |(y \lfloor\!\lfloor x)$.

<u>Case (iii)(c)4.</u> $z = a$. Similar to the next case.

<u>case (iii)(c)5.</u> $z = az'$. To prove $(az'|y) \lfloor\!\lfloor x \sqsubseteq az'|(y \lfloor\!\lfloor x)$. We use an induction on y:

<u>Case (iii)(c)5.1.</u> $y = y_1 + y_2$. Then $(az'|(y_1+y_2)) \lfloor\!\lfloor x = (az'|y_1) \lfloor\!\lfloor x +$
$(az'|y_2) \lfloor\!\lfloor x \sqsubseteq az'|(y_1 \lfloor\!\lfloor x) + az'|(y_2 \lfloor\!\lfloor x) = az'|((y_1+y_2) \lfloor\!\lfloor x) =$
$(az')|(y \lfloor\!\lfloor x)$.

<u>Case (iii)(c)5.2.</u> $y = \tau$: $(az'|\tau) \lfloor\!\lfloor x = \delta \lfloor\!\lfloor x = \delta \sqsubseteq az'|(\tau \lfloor\!\lfloor x)$.

<u>Case (iii)(c)5.3.</u> $y = \tau y'$: $(az'|\tau y') \lfloor\!\lfloor x = (az'|y') \lfloor\!\lfloor x \sqsubseteq (az')|(y' \lfloor\!\lfloor x)$ (*)
$(az')|(y' \| x) =_{(*)} (az')|\tau(y' \| x) = (az')|(\tau y' \lfloor\!\lfloor x)$.
(Note the curious manœuvre in steps (*).)

<u>Case (iii)(c)5.4.</u> $y = b$: $(az'|b) \lfloor\!\lfloor x = ((a|b)z') \lfloor\!\lfloor x = (a|b)(z' \| x) =$
$(az')|(bx) = (az')|(b \lfloor\!\lfloor x)$.

<u>Case (iii)(c)5.5.</u> $y = by'$: $(az'|by') \lfloor\!\lfloor x = ((a|b)(z' \| y')) \lfloor\!\lfloor x =$
$(a|b)(z' \| y') \| x) = (a|b)(z' \| (y' \| x)) = (az')|b(y' \| x) =$
$az'|((by') \lfloor\!\lfloor x)$.

(d) Finally we prove $\ell_5 \sqsubseteq r_2 + r_5 + r_7 \sqsubseteq R$ (and by permuting x,y we have then also $\ell_6 \sqsubseteq r_2 + r_6 + r_7 \sqsubseteq R$) i.e.:

$$\boxed{x|(y \lfloor\!\lfloor z) \sqsubseteq (x|y) \lfloor\!\lfloor z + (x \lfloor\!\lfloor y)|z + x|(y|z).}$$

The proof is again by induction on $|x| + |y| + |z|$. We start with an induction on x:

Case (iii)(d)1. $x = x_1 + x_2$. Then $x|(y \lfloor\!\lfloor z) = x_1|(y \lfloor\!\lfloor z) + x_2|(y \lfloor\!\lfloor z) \sqsubseteq$

$(x_1|y) \lfloor\!\lfloor z + (x_1 \lfloor\!\lfloor y)|z + x_1|(y|z) + (x_2|y) \lfloor\!\lfloor z +$

$(x_2 \lfloor\!\lfloor y)|z + x_2|(y|z) = (x|y) \lfloor\!\lfloor z + (x \lfloor\!\lfloor y)|z + x|(y|z)$.

Case (iii)(d)2. $x = \tau$. Then $x|(y \lfloor\!\lfloor z) = \delta \sqsubseteq (x|y) \lfloor\!\lfloor z + (x \lfloor\!\lfloor y)|z + x|(y|z)$.

Case (iii)(d)3. $x = \tau x'$. Then $\tau x'|(y \lfloor\!\lfloor z) = x'|(y \lfloor\!\lfloor z) \sqsubseteq$

$(x'|y) \lfloor\!\lfloor z + (x' \lfloor\!\lfloor y)|z + x'|(y|z) =$

$(\tau x'|y) \lfloor\!\lfloor z + (x' \lfloor\!\lfloor y)|z + \tau x'|(y|z) \sqsubseteq$

$(\tau x'|y) \lfloor\!\lfloor z + (x' \| y)|z + \tau x'|(y|z) =$

$(\tau x'|y) \lfloor\!\lfloor z + \tau(x' \| y)|z + \tau x'|(y|z) =$

$(\tau x'|y) \| z + (\tau x' \lfloor\!\lfloor y)|z + \tau x'|(y|z) =$

$(x|y) \lfloor\!\lfloor z + (x \lfloor\!\lfloor y)|z + x|(y|z)$.

Case (iii)(d)4. $x = a$: similar to the next case.

Case (iii)(d)5. $x = ax'$. To prove:

$$\boxed{(*) \quad ax'|(y \lfloor\!\lfloor z) \sqsubseteq (ax'|y) \lfloor\!\lfloor z + (ax' \lfloor\!\lfloor y)|z + ax'|(y|z).}$$

Subinduction to y: write $y = (\tau) + \sum c_i + \sum b_j y'_j + \sum \tau y''_\ell$. Clearly $ax'|(y\|z)$ can be decomposed as a sum analogous to the sum expression for y. Each of these summands of $ax'|(y \lfloor\!\lfloor z)$ will now be proved to be $\sqsubseteq$ the RHS of (*).

Case (iii)(d)5.1. Summands $b_j y'_j$: $(ax')|(b_j y'_j \lfloor\!\lfloor z) =$ (by statement (ii) of this theorem) $(ax'|b_j y'_j) \lfloor\!\lfloor z \sqsubseteq (ax'|y) \lfloor\!\lfloor z \sqsubseteq$ RHS(*).

Case (iii)(d)5.2. Summands c_i: as the previous case.

Case (iii)(d)5.3. Summand τ: $ax'|(\tau \lfloor\!\lfloor z) = ax'|\tau z = ax'|z = (ax' \lfloor\!\lfloor \tau)|z$ since $ax' = ax' \lfloor\!\lfloor \tau$ by Proposition 6.

Case (iii)(d)5.4. Summands $\tau y''_\ell$ (for convenience we drop the subscript ℓ and write $y = \tau y'' + y^*$):

Now $ax' \mid (\tau y'' \mathbin{\lfloor\!\lfloor} z) = ax' \mid \tau(y'' \| z) = ax' \mid (y'' \| z) =$

$ax' \mid (y'' \mathbin{\lfloor\!\lfloor} z + z \mathbin{\lfloor\!\lfloor} y'' + y' \mid z) =$

$ax' \mid (y'' \mathbin{\lfloor\!\lfloor} z) + ax' \mid (z \mathbin{\lfloor\!\lfloor} y'') + ax' \mid (y'' \mid z) \sqsubseteq$ (ind. hyp.)

$(ax' \mid y'') \mathbin{\lfloor\!\lfloor} z + (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z + ax' \mid (y'' \mid z) +$

$(ax' \mid z) \mathbin{\lfloor\!\lfloor} y'' + (ax' \mathbin{\lfloor\!\lfloor} z) \mid y'' + ax' \mid (y'' \mid z) + ax' \mid (y'' \mid z) =$

(Here the first summand equals the fifth by (ii) of this theorem, and likewise the second equals the fourth.)

$= (ax' \mid y'') \mathbin{\lfloor\!\lfloor} z + (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z + ax' \mid (y'' \mid z) =$

$(ax' \mid y'') \mathbin{\lfloor\!\lfloor} z + (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z + ax' \mid (\tau y'' \mid z) \sqsubseteq$

$(ax' \mid y) \mathbin{\lfloor\!\lfloor} z + (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z + ax' \mid (y \mid z)$.

This matches the RHS of (*) except for the second summand. So it remains to prove:

$$\boxed{\text{If } y = \tau y'' + y^*\text{, then } (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z \sqsubseteq (ax' \mathbin{\lfloor\!\lfloor} y) \mid z \quad (**)}$$

Proof of (**): induction on z.

<u>Case (iii)(d)5.4.1</u>. $z = z_1 + z_2$. Then $(ax' \mathbin{\lfloor\!\lfloor} y'') \mid (z_1 + z_2) = (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z_1 + (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z_2 \sqsubseteq (ax' \mathbin{\lfloor\!\lfloor} y) \mid z_1 + (ax' \mathbin{\lfloor\!\lfloor} y) \mid z_2 = (ax' \mathbin{\lfloor\!\lfloor} y) z$.

<u>Case (iii)(d)5.4.2</u>. $z = \tau$: $(ax' \mathbin{\lfloor\!\lfloor} y'') \mid \tau = \delta \sqsubseteq$ RHS(**).

<u>Case (iii)(d)5.4.3</u>. $z = \tau z'$: $(ax' \mathbin{\lfloor\!\lfloor} y'') \mid (\tau z') = (ax' \mathbin{\lfloor\!\lfloor} y'') \mid z' \sqsubseteq (ax' \mathbin{\lfloor\!\lfloor} y) \mid z' = (ax' \mathbin{\lfloor\!\lfloor} y) \mid (\tau z')$.

<u>Case (iii)(d)5.4.4</u>. $z = b$: $(ax' \mathbin{\lfloor\!\lfloor} y'') \mid b = a(x' \| y'') \mid b = (a \mid b)(x' \| y'')$.

Now $x' \| y = x' \| (\tau y'' + y^*) = x' \mathbin{\lfloor\!\lfloor} (\tau y'' + y^*) + (\tau y'' + y^*) \mathbin{\lfloor\!\lfloor} x' + x' \mid (\tau y'' + y^*) = \tau(y'' \| x') + T$.

So: $(ax' \mathbin{\lfloor\!\lfloor} y) \mid b = (a \mid b)(x' \| y) = (a \mid b)(\tau(y'' \| x') + T) \underset{\uparrow}{=}$ $(a \mid b)(\tau(y'' \| x') + T) + (a \mid b)(y'' \| x')$. Here "$\underset{\uparrow}{=}$" is an application of the third τ-law, T3. Therefore $(ax' \mathbin{\lfloor\!\lfloor} y'') \mid b = (a \mid b)(x' \| y'') \sqsubseteq (ax' \mathbin{\lfloor\!\lfloor} y) \mid b$.

<u>Case (iii)(d)5.4.5</u>. $z = bz'$: similar.

This ends the proof of induction step (iii), and thereby of the theorem. □

REFERENCES

[1] DE BAKKER, J.W. & J.I. ZUCKER, Processes and the denotational semantics of concurrency, Information and Control, Vol.54, No.1/2, 70-120, 1982.

[2] BERGSTRA, J.A. & J.W. KLOP, Process algebra for communication and mutual exclusion, Report IW 218/83, Mathematisch Centrum, Amsterdam 1983.

[3] BERGSTRA, J.A. & J.W. KLOP, An abstraction mechanism for process algebras, Report IW 231/83, Mathematisch Centrum, Amsterdam 1983.

[4] BERGSTRA, J.A. & J.W. KLOP, Algebra of Communicating Processes, Report IW 2../84, Centrum voor Wiskunde en Informatica, Amsterdam 1984.

[5] BERGSTRA, J.A. & J.V.TUCKER, Top-down design and the Algebra of Communicating Processes, Report CS-R8401 , Centrum voor Wiskunde en Informatica, Amsterdam 1984.

[6] BROOKES, S.D. & W.C. ROUNDS, Behavioural equivalence relations induced by programming logics, Proceedings 10th ICALP, Barcelona 1983, Springer LNCS 154 (ed. J. Díaz), 97-108.

[7] DERSHOWITZ, N., Orderings for term-rewriting systems, Theoretical Computer Science 17 (1982), 279-301.

[8] DERSHOWITZ, N., A note on simplification orderings, Information Processing Lett.9(5)(1979)212-215.

[9] GRAF, S. & J. SIFAKIS, A modal characterization of observational congruence on finite terms of CCS, to appear in Proceedings 11th ICALP, Antwerpen 1984.

[10] HENNESSY, M., A term model for synchronous processes, Information and Control, Vol.51, No.1 (1981), 58-75.

[11] HOARE, C.A.R., A model for communicating sequential processes, in: "On the construction of programs" (eds. R.M. McKeag and A.M. McNaghton), Cambridge University Press (1980), 229-243.

[12] MILNER, R., A Calculus for Communicating Systems, Springer LNCS 92, 1980.

[13] WINSKEL, G., Synchronisation trees, Proceedings 10th ICALP (ed. J. Díaz) Barcelona 1983, Springer LNCS 154, 695-711.