Theoretical Computer Science 37 (1985) 77–121
North-Holland

ALGEBRA OF COMMUNICATING PROCESSES WITH ABSTRACTION

J.A. BERGSTRA and J.W. KLOP

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

Communicated by E. Engeler
Received February 1984
Revised October 1984

Abstract. 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.

Keywords. Concurrency, communicating processes, internal actions, process algebra, bisimulation, process graph, handshaking, terminating rewrite rules, recursive path ordering.

1980 Mathematics Subject Classification. 68B10, 68C01, 68D25, 68F20.

1982 C.R. Categories. F.1.1, F.1.2, F.3.2, F.4.3.

Contents

Introduction

ACP, Algebra of Communicating Processes

In [2] we have introduced ACP (Algebra of Communicating Processes). ACP is an equational specification of process cooperation, aiming at an algebraic theory of processes; more specifically, ACP gives an equational framework for asynchronous process cooperation via synchronous communication. As an axiom system, it consists of the left column of Table 2 below. For a more extensive motivation of ACP as well as a discussion of related approaches, we refer to [2]. Here we will only mention that ACP is derived from Milner's Calculus of Communicating Systems (CCS); a discussion of the differences, both in the technical sence, as regards the signature, and in the methodological sense, is again contained in [2]. We will discuss two of these differences here: CCS has prefix multiplication (atomic process "a" and process q yield $a \cdot q$) whereas ACP admits general multiplication (processes p, q yield $p \cdot q$, the sequential composition). This is important for the expressive power: it is not hard to prove that several recursively defined processes have finite recursive definitions in terms of general multiplication, but not in terms of prefix multiplication. Now this adoption of general multiplication brings with it the introduction of a constant δ for deadlock: namely consider the process $p \cdot q$ where $p = \partial_{\{a,b\}}(a \| b)$, that is: the communication of steps a, b (encapsulated by $\partial_{\{a,b\}}$). Now if a, b cannot communicate, execution of $p \cdot q$ will not reach q. So p is a process which 'blocks' q. Indeed, in the ACP formalism the consequence of $a | b = \delta$ (a, b do not communicate) is that $p = \delta$ and now $p \cdot q = \delta \cdot q = \delta$. In prefix multiplication, $\partial_{\{a,b\}}(a \| b) \cdot q$ would not be a well-formed expression.

One of the aims of ACP is to keep track of the various models ('process algebras') which this axiomatisation has, rather than fixing a model right away as is done in CCS or related work in Hoare's CSP. Models of ACP can be given as projective limits of process algebras consisting of finite processes (see [4]), or as metrical completions of such process algebras (see [1]), or via process graphs. The latter method starts with a suitable domain of process graphs, i.e., rooted multidigraphs with edges labeled by atomic actions. Already here, there is a great variety of possibilities, as to the choice of an upper bound for the branching degrees, the cardinality of the node sets, etc. For instance, one may restrict the attention to regular process graphs, or, as is done below, to finite process graphs without cyclic paths. Having such a domain of process algebras, a suitable equivalence relation is divided out, e.g., bisimulation ($\leftrightarrows$). This notion derives from Milner's notion of strong equivalence on synchronisation trees (see [12]). For a proof-theoretical analysis of ACP, yielding results such as consistency of the axioms and an elimination property, we refer to [2].

ACP$_\tau$, *Algebra of Communicating Processes with abstraction*

ACP as briefly discussed above does not address the problem of abstraction ('hiding'), i.e., it does not deal with the so-called τ-steps (invisible or silent steps) of Milner. Now, ACP$_\tau$ is an extension of ACP which does take the presence of τ-steps into account. As an axiom system, ACP$_\tau$ is displayed in Table 2 below; it consists of ACP, in the left column, together with Milner's well-known 'τ-laws' T1-3. What is new in ACP$_\tau$ as compared with CCS is that ACP$_\tau$ also specifies the behaviour of τ-steps in relation with the communication merge operator "$|$", an operator which is not present in CCS but which is vital for giving a finite axiomatisation of merge "$\|$" as ACP does. (This is rather sensitive: e.g., $\tau a | b$ must yield the same process as $(\tau a + a) | b$, obtained by application of the τ-law $\tau a = \tau a + a$. Indeed it does: $(\tau a + a) | b = \tau a | b + a | b = a | b + a | b = a | b = \tau a | b$. Also, care had to be taken that the τ-laws are 'compatible' with the other auxiliary operator $\lfloor\!\lfloor$; see similar Examples 2.22.)

Another new feature in the treatment of τ-steps in ACP$_\tau$ is that in CCS communication between two atomic steps a, b yields a τ-step at once (if a, b are communication 'partners'). In ACP$_\tau$ the abstraction act is separated from the communication: abstraction is executed by a special operator τ_I. So $a | b$ is not τ right away, but first yields an internal (but still 'visible') step, say i, which later can be abstracted, i.e., renamed into τ: $\tau_{\{i\}}(a | b) = \tau_{\{i\}}(i) = \tau$. This separation of communication and abstraction turns out to be vital since *recursion and abstraction do not commute.* E.g., if processes X, Y are defined by recursion equations $X = aX$, $Y = bY$, then with $a | b = \tau$ we would get, for the parallel process $Z = \partial_{\{a,b\}}(X \| Y)$, the recursion equation $Z = \tau Z$. The problem is that in the presence of the τ-laws T1-3, such 'τ-guarded' recursion equations have no unique solution, i.e., Z is underspecified; indeed, every $Z = \tau p$ is a solution of $Z = \tau Z$ for arbitrary p. Using the τ_I-operator, the intended Z (i.e., the process "τ^ω") is easily defined, namely by putting $a | b = i$ and

$$Z = \tau_{\{i\}}\partial_{\{a,b\}}(X \| Y).$$

This matter is not pursued in the present paper, but these remarks may serve as a motivation for τ_I. In the present paper we are not concerned with explorations of the expressive power of ACP$_\tau$ but simply with introducing this system and proving some fundamental theorems about it so that it can serve as a firm basis for further explorations.

Summary of results

In Section 1, the signature of ACP$_\tau$ and the axioms of ACP$_\tau$ are given. This signature extends that of ACP by the presence of τ_I and τ; all axioms involving them are in the right column of Table 2.

In Section 2 we give a simple model for ACP$_\tau$, consisting of finite process graphs without cycles, modulo an equivalence relation $\leftrightarrows_{r\tau}$ called *rooted τ-bisimulation.* Here, 'τ-bisimulation' ($\leftrightarrows_\tau$ in our notation) coincides with Milner's well-known notion of observational equivalence, at least for finite processes. A problem with observational equivalence, or τ-bisimulation, is that it is not a congruence w.r.t. the operations $+$ and $\lfloor\!\lfloor$ (the typical example is that while $\tau a \leftrightarrows_\tau a$, $b \leftrightarrows_\tau b$, one has $\tau a + b \not\leftrightarrows_\tau a + b$). Therefore, we consider a mild variant of it, $\leftrightarrows_{r\tau}$, which is a congruence w.r.t. all operators. For an algebraic approach it seems essential to work with congruences; thus we can take the quotient algebra of the domain of finite acyclic processes modulo $\leftrightarrows_{r\tau}$, and this algebra is proved in Theorem 2.23 to be isomorphic to the initial algebra of ACP_τ. Otherwise said, ACP_τ is a complete axiomatisation for this process algebra. (The completeness of the τ-laws for finite processes was first proved by Milner, as stated in [12], for the smaller signature as favoured by CCS.) This proof, which also entails the conservativity of ACP_τ over ACP (i.e., no unwanted identifications are caused by the extension from ACP to ACP_τ), makes a typical use of the underlying graph domain. An important ingredient in the proof is the use of some very simple transformations of process graphs. These transformations tend to normalize a process graph; a key fact (Theorem 2.12) states that each $r\tau$-bisimulation equivalence class contains a unique normal graph. Another important fact, of independent interest, used in the proof is the Elimination Theorem 2.20 stating that the 'defined' operators $\|$, $\lfloor\!\lfloor$, $|$, ∂_H, τ_I can all be eliminated (in a finite process) in favour of the 'basic constructors' $+$ and $\cdot$. Since the proof of the Elimination Theorem requires quite some work, it is contained in Appendix A. The method used to prove the termination of the rewrite rules, which tend to eliminate the defined operators, is that of the 'recursive path ordering' as described by Dershowitz, based on Kruskal's Tree Theorem. In proofs like this termination proof, it is important to have an axiomatisation as ACP_τ gives, which lends itself to simple rewrite rules. Appendix A gives next to the actual application of the recursive path ordering method to the ACP_τ-termination problem, a (mostly notational) restatement of the r.p.o. method which we find helpful when we actually use it for a rather complicated rewrite system as the one under consideration.

Section 3 proves the Expansion Theorem for ACP_τ (Theorem 3.9). This theorem, first proved by Milner [12] for CCS, was proved in [5] for ACP. The extension to ACP_τ turns out to be nontrivial, but the theorem, which is indispensable for breaking down merge expressions $x_1 \| \ldots \| x_k$, fortunately holds in exactly the same form as for ACP:

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

where X_k^i stands for 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. (Note that the auxiliary operators $\lfloor\!\lfloor, |$ make a succinct formulation possible.)

Finally, in Appendix B we prove by a straightforward induction on term formation the associativity of merge "$\|$"; by a different, indirect, method this is also done in Section 3 (Corollary 3.8) but we have preferred also to include the proof in Appendix B because it is entirely algebraical, using the axioms of ACP_τ, thus demonstrating their ease in computations (the second half of the proof in Appendix B uses a complicated simultaneous induction, though), and because it proves more, viz. several identities which are of independent interest.

We conclude this introduction with some remarks about related literature (for a more comprehensive comparison, see [2]).

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 $+$, $\|$, $a\cdot$ 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 $\cdot$ is present in Hoare's CSP [11] as ";" and in [1] as "$\circ$". We refer to Graf and Sifakis [9] for a proof-theoretic discussion of the τ-laws. Brookes and Rounds [6] give an explicit description of bisimulation modulo τ on finite graphs.

1. The axiom system ACP_τ

Let A be a finite set of atomic actions, containing a constant δ, and let $\cdot\,|\,\cdot : A\times A\to A$ 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 given in 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).

Table 1.

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

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 is the axiom system ACP (without τ). In Table 2, "a" varies over A.

The axioms T1, T2, T3 are the 'τ-laws' from Milner [12].

Notation: Often we will write xy instead of $x\cdot y$.

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

Table 2. ACP$_\tau$.

$x+y=y+x$	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		
$a\mid b=b\mid a$	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		
$a\lfloor\!\lfloor x=ax$	CM2	$\tau\lfloor\!\lfloor x=\tau x$	TM1
$(ax)\lfloor\!\lfloor y=a(x\parallel y)$	CM3	$(\tau x)\lfloor\!\lfloor y=\tau(x\parallel y)$	TM2
$(x+y)\lfloor\!\lfloor z=x\lfloor\!\lfloor z+y\lfloor\!\lfloor z$	CM4	$\tau\mid x=\delta$	TC1
$(ax)\mid b=(a\mid b)x$	CM5	$x\mid\tau=\delta$	TC2
$a\mid(bx)=(a\mid b)x$	CM6	$(\tau x)\mid y=x\mid y$	TC3
$(ax)\mid(by)=(a\mid b)(x\parallel y)$	CM7	$x\mid(\tau y)=x\mid y$	TC4
$(x+y)\mid z=x\mid z+y\mid z$	CM8		
$x\mid(y+z)=x\mid y+x\mid z$	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
$\partial_H(xy)=\partial_H(x)\cdot\partial_H(y)$	D4	$\tau_I(xy)=\tau_I(x)\cdot\tau_I(y)$	TI5

2. The model of finite acyclic process graphs for ACP$_\tau$

A *process graph* over A_τ is a rooted, directed multigraph such that every node is accessible from the root and whose edges are labeled by elements from A_τ. A process graph is *finite* if it has finitely many edges and nodes; it is *acyclic* when it contains no cyclic path, i.e., there are no edges $h_i = s_i \xrightarrow{l_i} s_{i+1}$ $(i<k,\ l_i\in A_\tau)$ and nodes s_j $(j\leq k)$ such that

$$s_0 \xrightarrow[h_0]{l_0} s_1 \xrightarrow[h_1]{l_1} \cdots \xrightarrow[h_{k-1}]{l_{k-1}} s_k = s_0 \quad (k\geq 1).$$

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 *δ-normal* process graph. A process graph $g\in G$ is δ-normal if whenever an edge

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

occurs in g, then the node s as outdegree 1 and the node t has outdegree 0. In anthropomorphic terminology, let us say that an edge

$$\circled{s} \to \circled{t} \text{ is an } \textit{ancestor of } \circled{s'} \to \circled{t'}$$

if it is possible to move along edges from t to s'; likewise, the latter edge will be called a *descendant* of the former. Edges having the same begin node are *brothers*. 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) *nondeterministic δ-removal* is the elimination of a δ-edge having at least one brother,

(2) *δ-shift* of a δ-edge

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

in g consists of deleting this edge, creating a fresh node t' and adding the edge

$$\circled{s} \xrightarrow{\delta} \circled{t'}.$$

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 *δ-normal form* of g. It is understood that pieces of the graph which have become disconnected from the root, are discarded.

Example

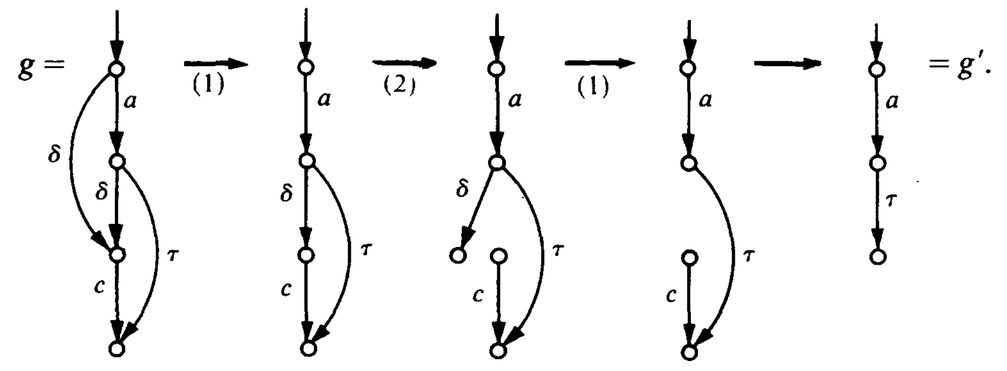


Fig. 1.

We can now define bisimulation between process graphs $g_1, g_2 \in G$. First some preliminary notions: a *trace* 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

$$\circled{s_0} \xrightarrow[h_0]{l_0} \circled{s_1} \xrightarrow[h_1]{l_1} \cdots \xrightarrow[h_{k-1}]{l_{k-1}} \circled{s_k} \quad (k \geq 0)$$

where the s_i are nodes of g, the h_i are edges between s_i and s_{i+1}, and each $l_i \in A_\tau$

is the label of edge h_i. (The h_i are needed because we work with multigraphs.) The trace $trace(\pi)$ associated to this path π is just $l_0 l_1 \dots l_{k-1}$.

2.1. Definition. A *bisimulation modulo* τ (or τ-*bisimulation*) between finite acyclic process graphs g_1 and g_2 is a relation R on $\text{NODES}(g_1) \times \text{NODES}(g_2)$ 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 $e(trace(\pi_1)) = e(trace(\pi_2))$ (see Fig. 2(a)).

(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 $e(trace(\pi_1)) = e(trace(\pi_2))$ (see Fig. 2(b)).

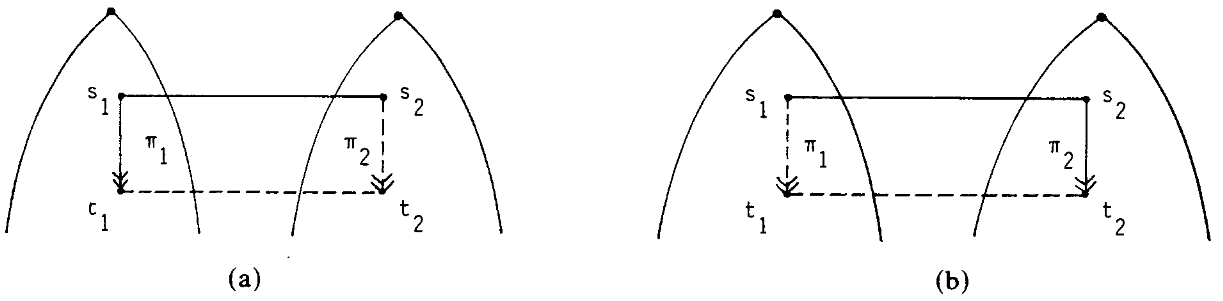


Fig. 2.

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

Notation: $g_1 \leftrightarrows_\tau g_2$.

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

Strictly speaking we should say that R as in Definition 2.1 is a τ-bisimulation from g_1 to g_2 rather than between g_1, g_2. Note that if R is a τ-bisimulation from g_1 to g_2, the converse relation R^{-1} (defined by $(s, t) \in R^{-1} \Leftrightarrow (t, s) \in R$) is a τ-bisimulation from g_2 to g_1.

2.2. Definition. Let $g_1, g_2 \in G$ be in δ-normal form. A *rooted bisimulation 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: $g_1 \leftrightarrows_{r\tau} g_2$.

2.3. Definition. Let $g_1, g_2 \in G$ with δ-normal forms g_1', g_2' respectively. Then $g_1 \leftrightarrows_{r\tau} g_2$ if $g_1' \leftrightarrows_{r\tau} g_2'$.

2.4. Examples

$a\tau b\delta \leftrightarrows_{r\tau} ab\delta$ (Fig. 3(a)),

$ab \leftrightarrows_{r\tau} a\tau(\tau b+\tau\tau b)$ (Fig. 3(b)),

$a(\tau b+b) \leftrightarrows_{r\tau} ab$ (Fig. 3(c)),

$c(a+b) \leftrightarrows_{r\tau} c(\tau(a+b)+a)$ (Fig. 3(d)).

For a negative example, see Fig. 3(e). The heavy line denotes where it is not possible to continue a construction of the bisimulation.

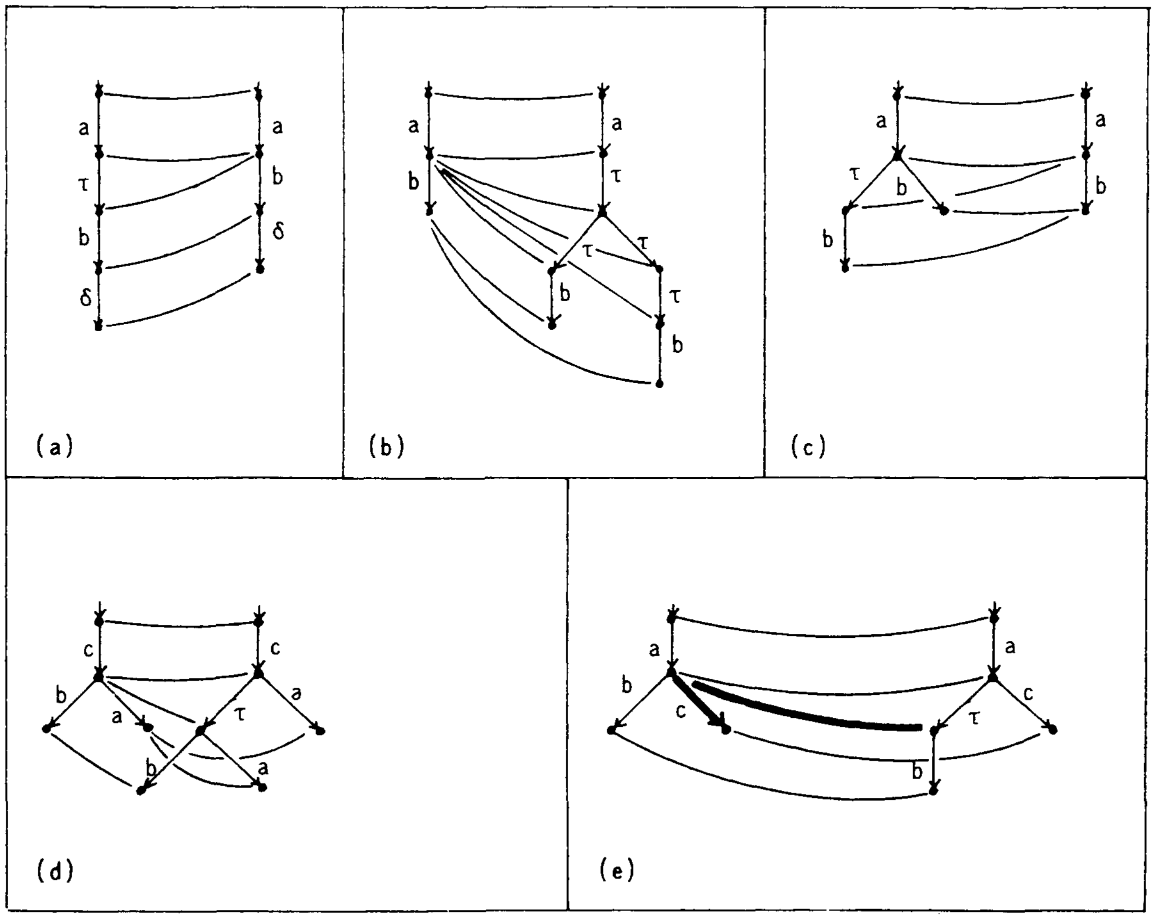


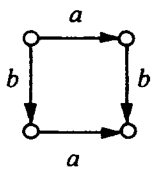
Fig. 3.

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.)

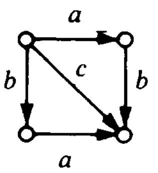
(1) The *sum* g_1+g_2 is the result of identifying the roots of g_1, g_2.

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

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

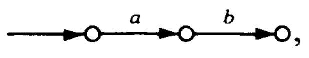


is a subgraph of the cartesian product graph, then the arrow $\circ\xrightarrow{c}\circ$ (where $c = a\,|\,b$) is inserted; result:



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

Example. Let $A_\tau = \{a, b, c, \tau, \delta\}$, where the only nontrivial communication is: $a\,|\,b = c$. Then, writing ab for the graph



we have: $ab\,\|\,bab\tau$ is the process graph as in Fig. 4(a).

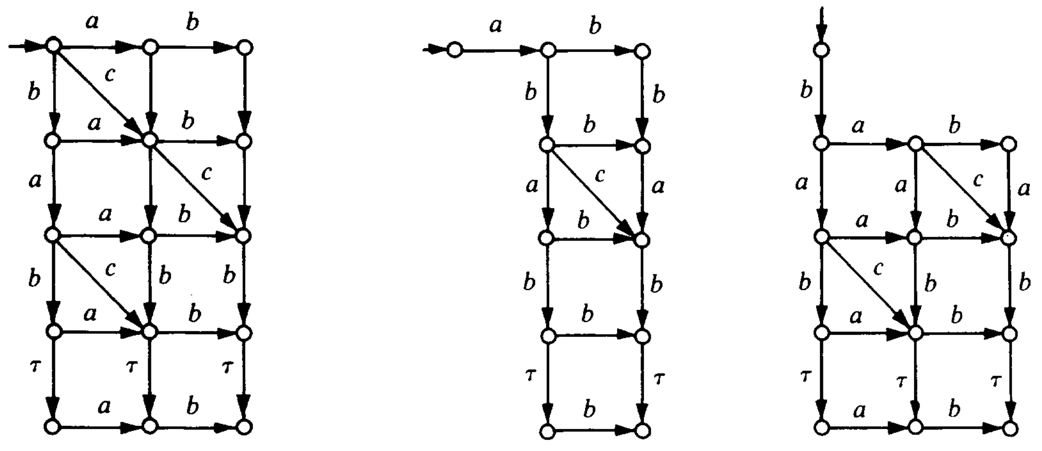


Fig. 4.

(4) The *left merge* $g_1 \mathbin{\|\!\lfloor} g_2$ 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.

Example: In the situation of the previous example we have $ab \mathbin{\|\!\lfloor} bab\tau$ as the graph in Fig. 4(b) and $bab\tau \mathbin{\|\!\lfloor} ab$ as in Fig. 4(c).

(*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 $g_1 \mathbin{\|\!\lfloor} g_2$ 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$_\tau$: $\tau\tau ax\,|\,\tau\tau\tau by = ax\,|\,by = (a\,|\,b)\cdot(x\,\|\,y)$.

We define:

$g_1\,|\,g_2 = \sum\{(t \to s)\cdot(g_1\,\|\,g_2)_s \mid t \to s$ is a maximal communication in $g_1\,\|\,g_2$ such that t can be reached from the root via a sequence of τ-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 $a \mid b = c$ be the only nontrivial communication. Then $g_1 \parallel g_2$ is as in Fig. 5(a) and $g_1 \mid g_2$ as in Fig. 5(b):

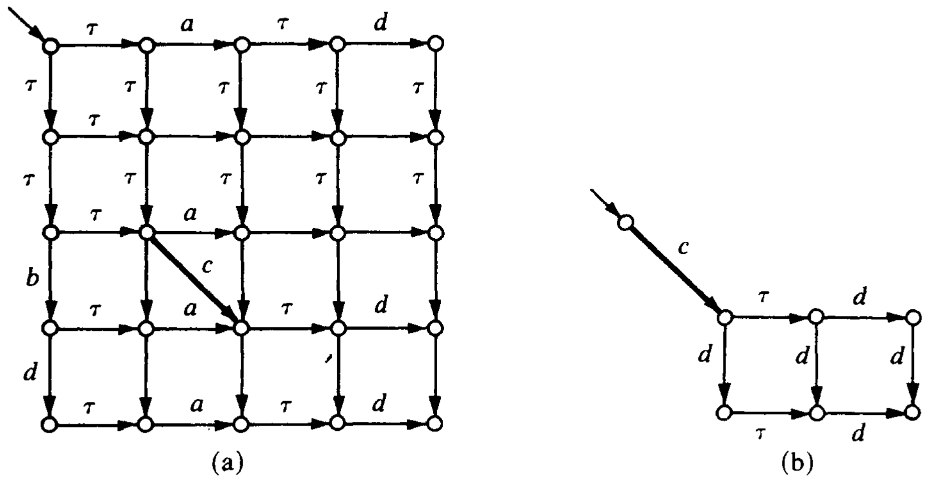


Fig. 5.

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

(ii) Let

$$g_1: \begin{tikzcd} {} \arrow[r] & \circ \arrow[r, shift left, "b"] \arrow[r, shift right] & \circ \arrow[r, "a"] & \circ \end{tikzcd} \quad \text{and} \quad g_2: \begin{tikzcd} {} \arrow[r] & \circ \arrow[r, shift left, "a"] \arrow[r, shift right] & \circ \arrow[r, "b"] & \circ \end{tikzcd},$$

where the only nontrivial communications are $a \mid a = a^0$ and $b \mid b = b^0$. Then $g_1 \parallel g_2$ and $g_1 \mid g_2$ are as in Fig. 6(a), (b) respectively:

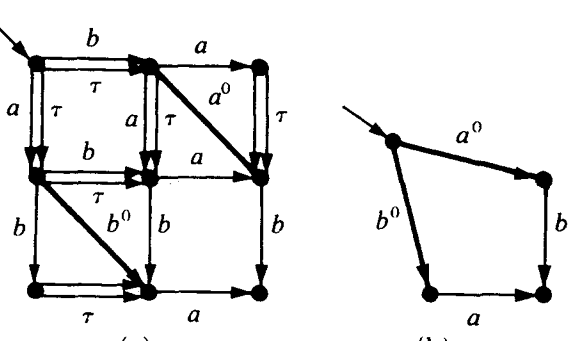


Fig. 6.

Using ACP$_\tau$ we calculate with terms corresponding to g_1, g_2:

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

$$= (b \mid a) \cdot (a \parallel b) + ba \mid b + a \mid ab + a \mid b = \delta + b^0 a + a^0 b + \delta = b^0 a + a^0 b.$$

(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, τ respectively.

This ends the definition of the structure $\mathcal{G} = G(+, \cdot, \parallel, \lfloor\!\lfloor, \mid, \partial_H, \tau_I)$. The domain of process graphs $\mathcal{G}$ itself is not yet a model of ACP$_\tau$ (e.g., $\mathcal{G} \not\models x + x = x$). However, we have the following theorem.

2.5. Theorem

(i) *Rooted τ-bisimulation* ($\leftrightarrows_{r\tau}$) *is a congruence on* $\mathcal{G}$.

(ii) $\mathcal{G}/\leftrightarrows_{r\tau}$ *is a model of* ACP$_\tau$.

Proof. (i) Let $g, g', h, h' \in G$. We want to show that

$$g \leftrightarrows_{r\tau} g' \ \&\ h \leftrightarrows_{r\tau} h' \ \Rightarrow\ g \parallel h \leftrightarrows_{r\tau} g' \parallel h'$$

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

Suppose, then, that S is an rτ-bisimulation between g, g' and T is an rτ-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 $g\parallel h$ and $g'\parallel h'$:

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

We *claim* that $S\times T$ is an rτ-bisimulation between $g\parallel h$ and $g'\parallel h'$.

Proof of the claim. (1) Let $(s_1,t_1)\xrightarrow{u}(s_1,t_2)$ be a 'horizontal step' in $g\parallel h$, 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 $g\parallel h$.

(2) Likewise for a 'vertical step' in $g\parallel h$.

(3) $(s_1,t_1)\xrightarrow{c}(s_2,t_2)$ is a 'diagonal step' (a communication step) in $g\parallel h$, 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 Fig. 7.

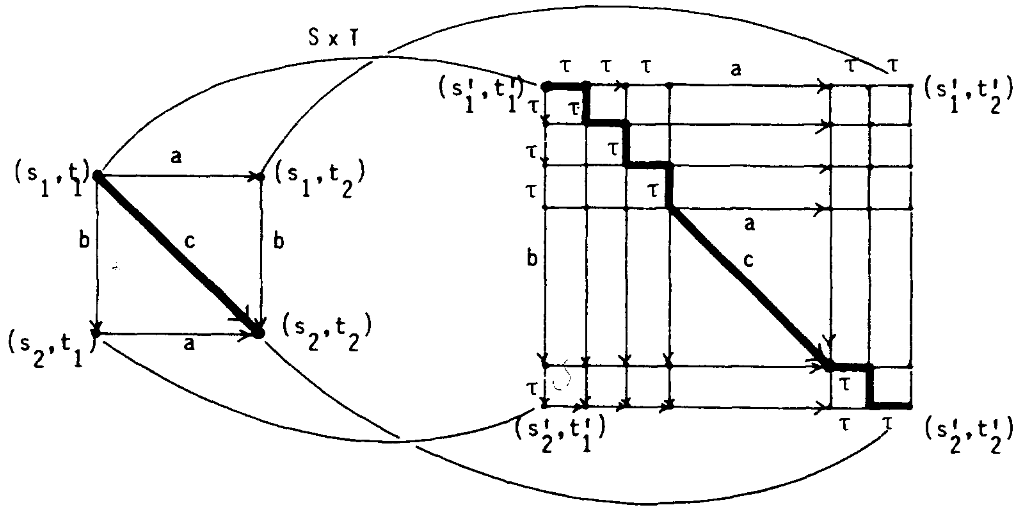


Fig. 7.

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

For the case of $\mid$ we use the same notation as above. We have to prove

$$g\mid h \leftrightarrows_{r\tau} g'\mid h'.$$

An $r\tau$-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 Fig. 8(a).

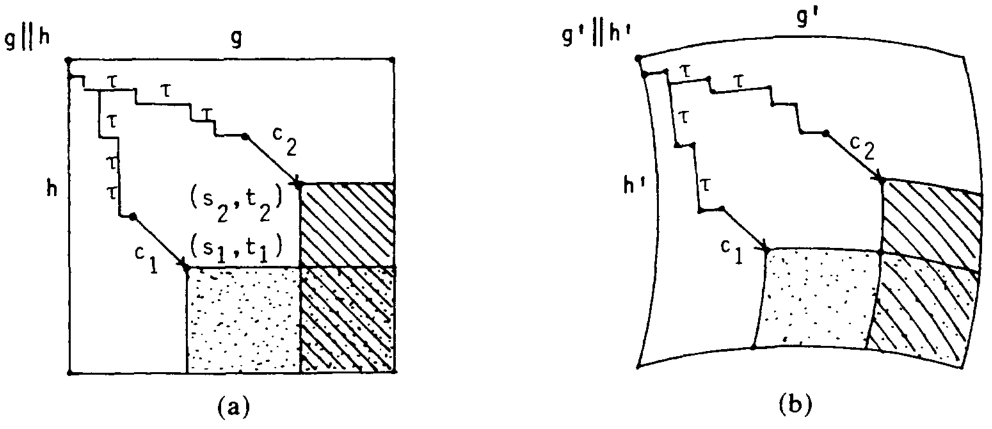


Fig. 8.

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 $S\times T$ 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_τ is tedious and routine. We will sketch some of it: the soundness of the axioms A1-7 is easy; e.g., associativity of $+$, $\cdot$ and commutativity of $+$ follow at once from properties of graphs; soundness of A3, $x+x=x$, is a simple consequence of properties of bisimulation and A6, 7 follow because the graphs in $\mathcal{G}$ are in δ-normal form. The axiom groups for ∂_H and τ_I are trivially sound. The τ-laws T1-3 are proved sound by constructing the appropriate $r\tau$-bisimulation. For an axiom as CM1 an $r\tau$-bisimulation between the graphs $g\,\|\,h$ and $g \;⌊\!⌊\; h+h \;⌊\!⌊\; g+g\,|\,h$ is constructed by connecting nodes in $g \;⌊\!⌊\; h$, $h \;⌊\!⌊\; g$, $g\,|\,h$ to corresponding nodes in $g\,\|\,h$, where 'corresponding' refers to the way in which $g \;⌊\!⌊\; h$, $h \;⌊\!⌊\; g$, $g\,|\,h$ are directly constructed from $g\,\|\,h$. Finally, axioms as CM2-9, TM1, 2, TC1-4 are easily dealt with. □

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 $s \in \text{NODES}(g)$. 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 Fig. 9(a), where $u \in A_\tau$. The u-edge at the left is called the *primary edge* of the arc. If, in Fig. 9(a), $n = m = 0$, the arc has the form as in Fig. 9(b) and is called of type I. If $n + m = 1$, the arc has the form as in Fig. 9(c) or (d) and is called of type II, III respectively. Arcs of type I, II, III are called *elementary arcs.*

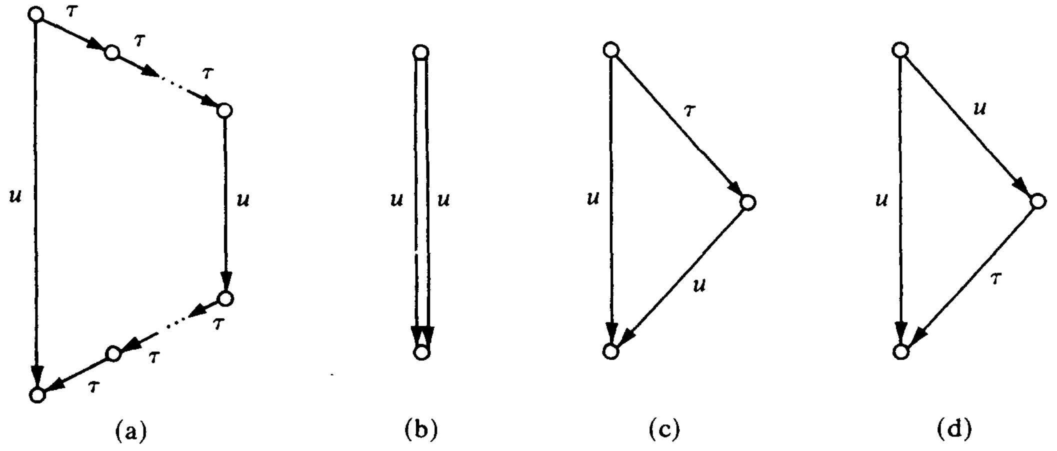


Fig. 9.

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 Fig. 10(a) may be replaced by that in Fig. 10(b):

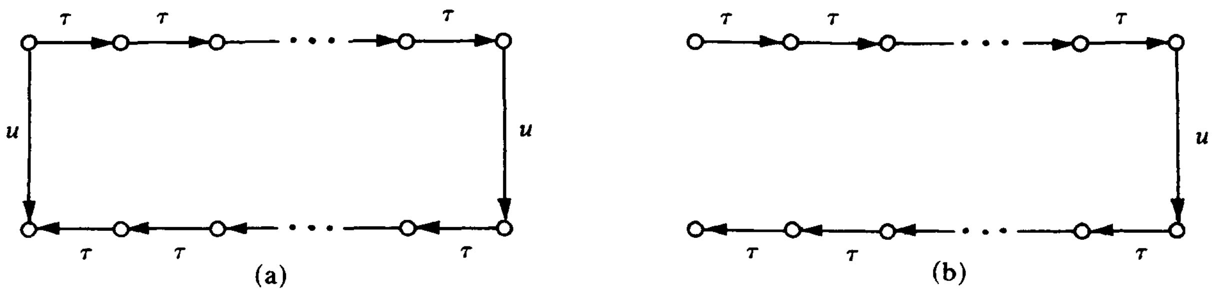


Fig. 10.

[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 $g \rightarrow g'$. The transitive reflexive closure of $\rightarrow$ is denoted by $\twoheadrightarrow$.

2.8. Example

(i)

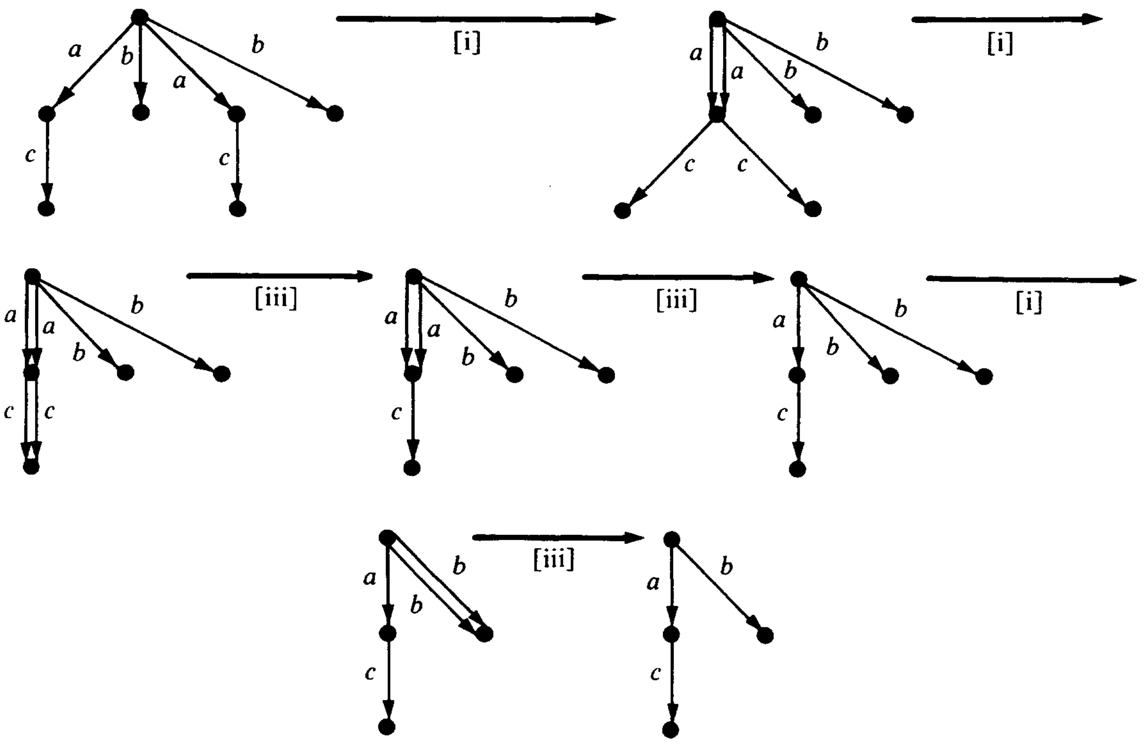


Fig. 11.

(ii)

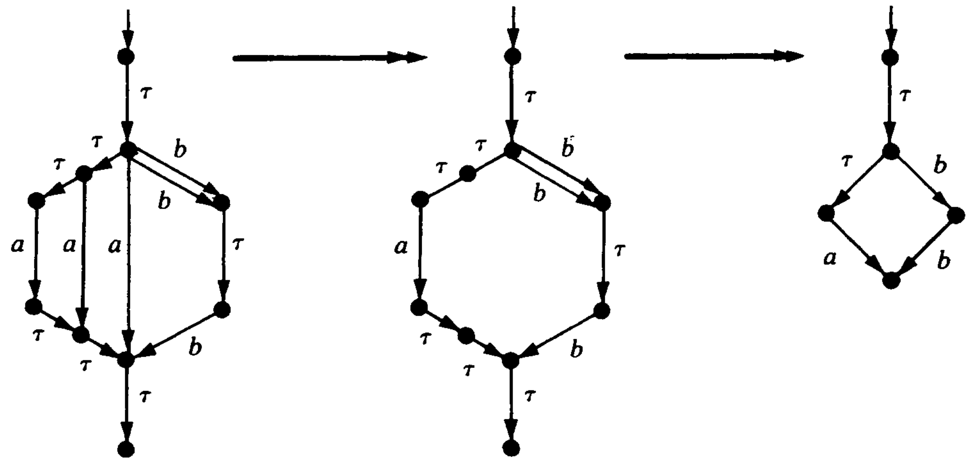


Fig. 12.

(iii)

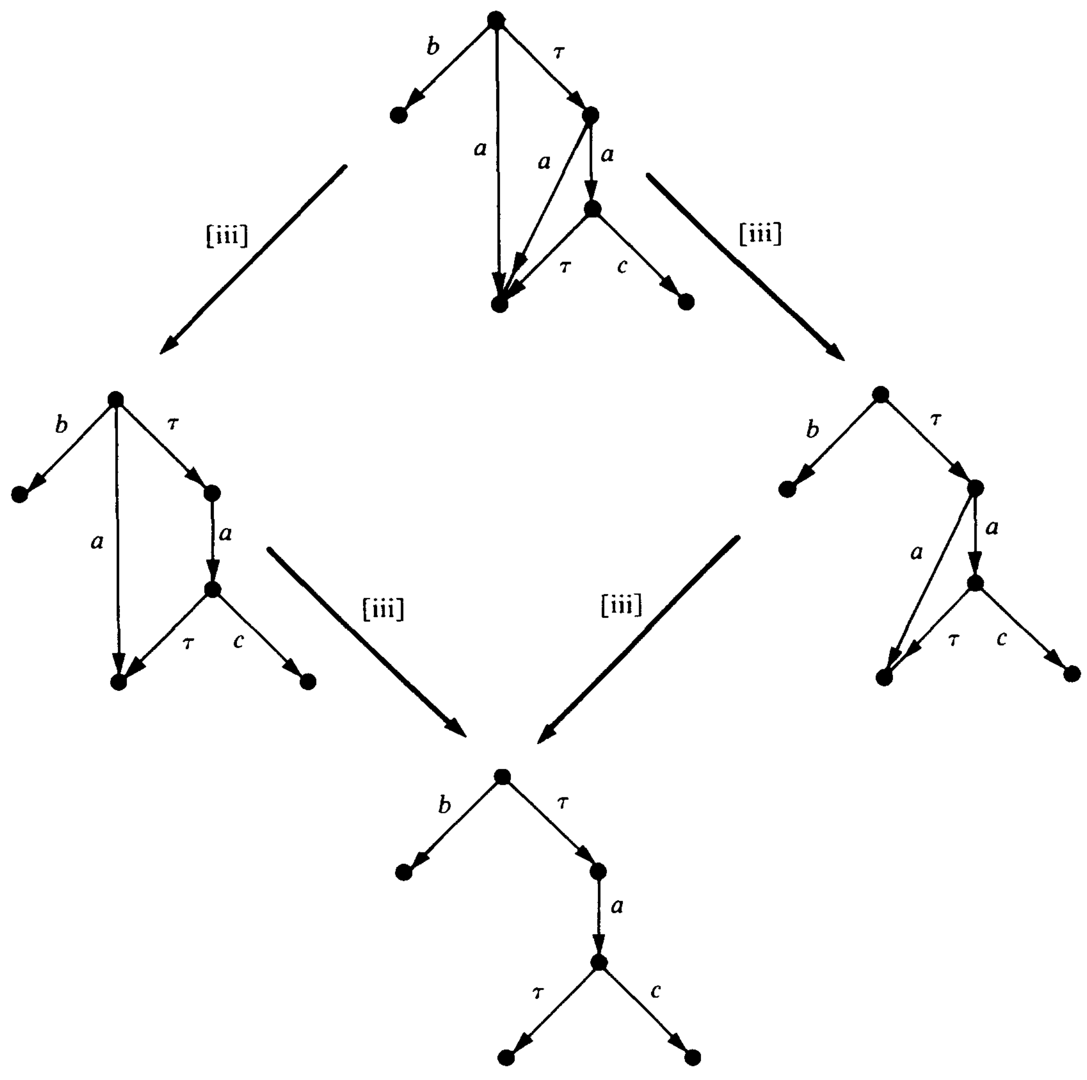


Fig. 13.

2.8.1. Remark. As the last example suggests, the process graph reduction $\twoheadrightarrow$ is in fact confluent (has the Church-Rosser property). A proof of the confluency can be obtained by the following trivial proposition (2.9) together with an analysis of the 'elementary diagrams' as in Example 2.8(iii) above, showing the weak confluency property: if $g_1 \to g_2$ and $g_1 \to g_3$, then there is a g_4 such that $g_2 \twoheadrightarrow g_4$ and $g_3 \twoheadrightarrow g_4$. Establishing this weak confluency directly, is rather complicated though; much easier is the following way, indicated in Remark 2.14. We will not need the confluency of the graph reductions in the sequel.

The following fact is trivial.

2.9. Proposition. *Every process graph reduction* $g_1 \to g_2 \to \cdots$ *must terminate eventually.*

Without the routine proof we state the 'soundness' of the reduction procedure $\twoheadrightarrow$ w.r.t. $\leftrightarrows_{r\tau}$.

2.10. Lemma. *Let $g_1, g_2 \in G$. Then $g_1 \twoheadrightarrow g_2$ implies $g_1 \leftrightarrows_{r\tau} g_2$.*

2.11. Definition. (i) Let $g \in G$ be in δ-normal form. Let R be an rτ-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 process graph depicted in Fig. 14 is not rigid since it admits the displayed nontrivial autobisimulation:

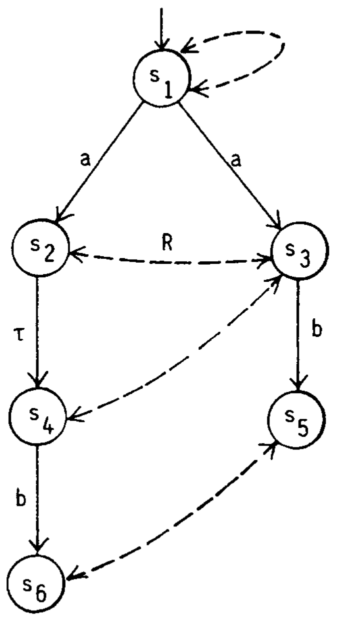


Fig. 14.

Here $R = \{(s_1, s_1), (s_2, s_3), (s_3, s_2), (s_4, s_3), (s_3, s_4), (s_5, s_6), (s_6, s_5)\}$.

2.11.2. Proposition. *If g, h are rigid and R is an rτ-bisimulation from g to h, then R is in fact a bijection from* NODES(g) *to* NODES(h).

Proof. Suppose there are $s_1, s_2 \in$ NODES(g), $t \in$ NODES(h) such that $(s_1, t) \in R$, $(s_2, t) \in R$. Now if R is a bisimulation from g to h, the converse of R, R^{-1}, is a bisimulation from h to g and the composition $R^{-1} \circ R$ is a bisimulation from g to g, i.e., an autobisimulation of g. Since $(s_1, s_2) \in R^{-1} \circ R$, it follows from the rigidity of g that $s_1 = s_2$. □

2.12. Theorem. (i) *Normal graphs are rigid.*

(ii) *If g_1, g_2 are normal process graphs and $g_1 \leftrightarrows_{r\tau} g_2$, then g_1 and g_2 must be identical.*

Proof. We will prove (i), (ii) simultaneously with induction on the size of the process graphs involved. To be precise: Let $|g|$ be the number of edges of g. Consider the statements:

$(i)_n$: if g is a normal process graph, and $|g| \leq n$, then g is rigid.

$(ii)_n$: if g_1, g_2 are normal process graphs such that $|g_1|, |g_2| \leq n$ and $g_1 \leftrightarrows_{r\tau} g_2$, then $g_1 = g_2$.

We will prove $(i)_n$ & $(ii)_n$ with induction on n. The basis, $(i)_1$ & $(ii)_1$, is trivial. (Note here that the one edge graph $\rightarrow\circ\xrightarrow{\tau}\circ$ is rigid.)

Induction step: suppose as induction hypothesis that $(i)_k$ & $(ii)_k$ holds for $k < n$. To prove: $(i)_n$ & $(ii)_n$. We first prove $(i)_n$.

So let g with $|g| = n$ be normal. Suppose, for a proof by contradiction, that g is not rigid; then there is an $r\tau$-autobisimulation of g relating unequal nodes s, t. By definition of $r\tau$-bisimulation, none of s, t can be the root of g. But then the subgraphs $(g)_s$, $(g)_t$ have at least two edges less than g, i.e., $|(g)_s|, |(g)_t| \leq n-2$. Clearly $(g)_s \leftrightarrows_\tau (g)_t$ and hence $\tau.(g)_s \leftrightarrows_{r\tau} \tau.(g)_t$. (Here $\tau.(g)_s$ results from prefixing a τ-edge to $(g)_s$.) Moreover $(g)_s$, $(g)_t$ are normal since they are subgraphs of a normal graph; and since s, t are non-root nodes, $(g)_s$, $(g)_t$ do not start with a 'deterministic' τ-step and hence also $\tau(g)_s$, $\tau(g)_t$ are normal.

Now, since $|\tau(g)_s|, |\tau(g)_t| \leq n-1$, we may apply the induction hypothesis and conclude that $\tau(g)_s = \tau(g)_t$ and hence $(g)_s = (g)_t$. But then g would admit a $\underset{[i]}{\rightarrow}$-step, contradicting the normality. Hence g is rigid and we have proved $(i)_n$.

Next we prove $(ii)_n$, using the induction hypothesis $(i)_k$ & $(ii)_k$ $(k < n)$ and $(i)_n$. So let g_1, g_2 be normal such that $g_1 \leftrightarrows_{r\tau} g_2$ and $|g_1|, |g_2| = n$. By $(i)_n$, the graphs g_1, g_2 are rigid. Let R be an $r\tau$-bisimulation from g_1 to g_2. By Proposition 2.11.2, R is a bijection from $\text{NODES}(g_1)$ to $\text{NODES}(g_2)$. Furthermore, we claim that R maps the edges of g_1 bijectively to those of g_2; more precisely:

Claim. (1) *If* $s \xrightarrow{u} t$, $u \in A_\tau$, *is an edge of* g_1 *and* $(s, s') \in R$, *then there is an edge* $s' \xrightarrow{u} t'$ *in* g_2 *for some* t' *with* $(t, t') \in R$.

(2) *Likewise vice versa.*

With the claim we are through, since R is then an isomorphism between labeled graphs. (Intuitively, this can easily be seen by noting that a process graph g without double edges can be considered as an algebraic structure, in the sense of model theory, with universe $\text{NODES}(g)$, a constant $\text{ROOT}(g)$ and binary relations $a, b, c, \ldots$, the labels of the edges of g.)

Proof of the Claim. (1) Let $s \xrightarrow{u} t$ be an edge as in (1) of the Claim. Let $(s, s') \in R$ (see Fig. 15). Suppose there is not a 'direct' step $s' \xrightarrow{u} t'$, $(t, t') \in R$ as we want. Then, since R is an $r\tau$-bisimulation, there is a path

$$s' \xrightarrow{\tau^n} s'' \xrightarrow{u} t'' \xrightarrow{\tau^m} t'$$

from s' to t', $(t, t') \in R$ (see Fig. 15) such that $n + m \neq 0$. Say $n \neq 0$ (the other case, $m \neq 0$, is similar). Going 'backwards' from s'' with R we find a node s''' in g_1 such that $s \xrightarrow{\tau^{n'}} s'''$, $(s''', s'') \in R$. Since $n \neq 0$ and because R is a bijection between the node sets of g_1, g_2, we have $s \neq s'''$, i.e., $n' \neq 0$. Likewise, the path $s'' \xrightarrow{u} t'' \xrightarrow{\tau^m} t'$ in g_2 is carried via R backwards to g_1 to yield a path

$$s''' \xrightarrow{\tau^k u \tau^l} t^*.$$

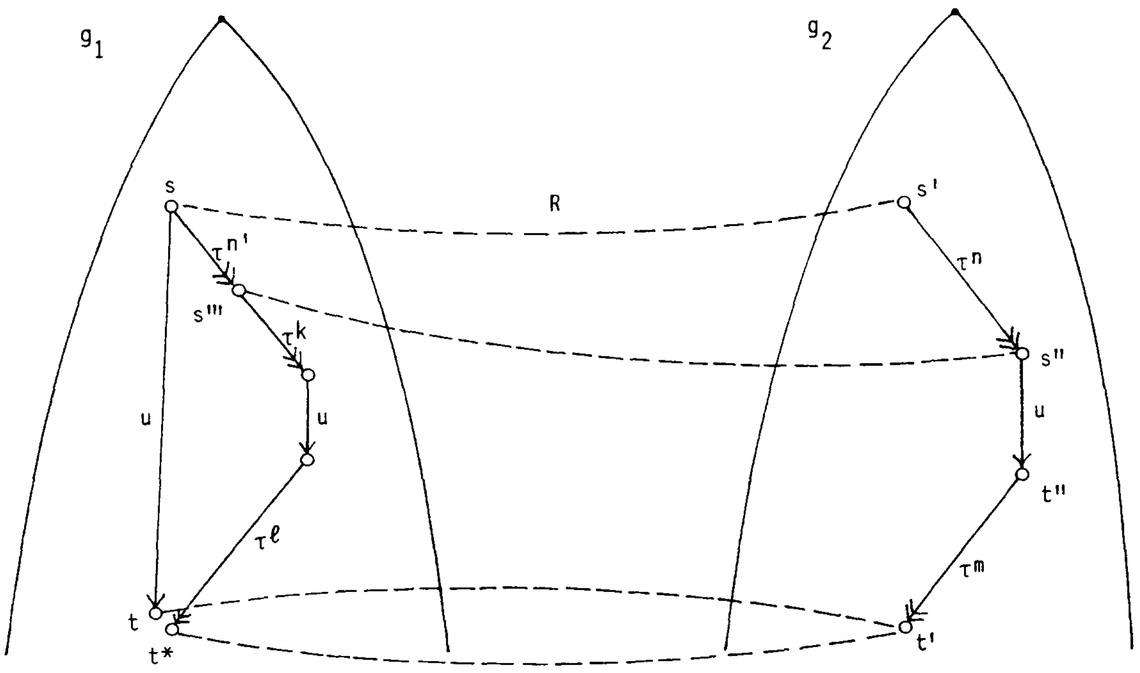


Fig. 15.

By the bijectivity of R, $t=t^*$. But then there is an arc in g_1, in contradiction with the normality of g_1. Hence, there is a direct step $s' \xrightarrow{u} t'$, $(t, t') \in R$.

Part (2) of the Claim is like (1), with g_1, g_2 interchanged. □

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

(i) $g_1 \leftrightarrows_{r\tau} g_2$,

(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. $\leftrightarrows_{r\tau}$, also $g_1' \leftrightarrows_{r\tau} g_2'$. By Theorem 2.12(ii) it follows that g_1' and g_2' are identical; hence (ii). From (ii) we trivially have (iii). 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 immediately follows from the termination property of the graph reductions (Proposition 2.9), together with Lemma 2.10 and Theorem 2.12(ii).

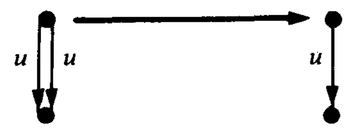
2.15. Corollary. *Let $g_1, g_2 \in \mathcal{G}$. Then $g_1 \leftrightarrows_{r\tau} g_2$ iff g_1, g_2 are convertible by means of the following reductions*:

[i] *sharing (as in Definition* 2.7*)*;

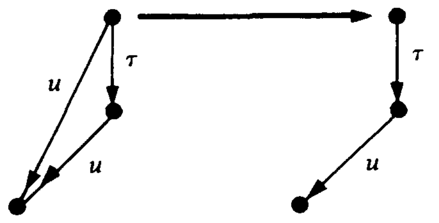
[ii] *removal of a non-initial deterministic τ-step (as in Definition* 2.7*)*;

The elementary arc reductions:

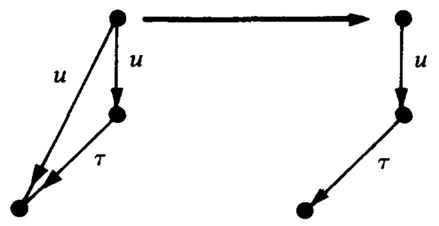
[iii]I



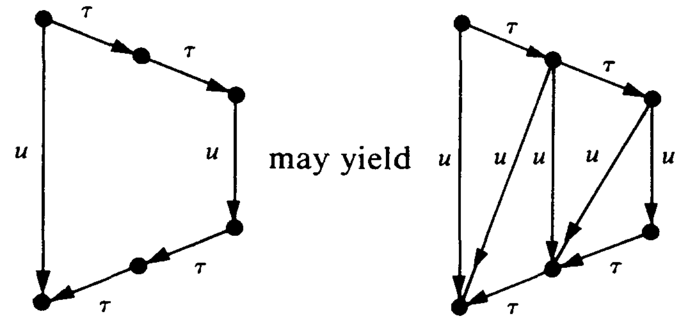
[iii]II



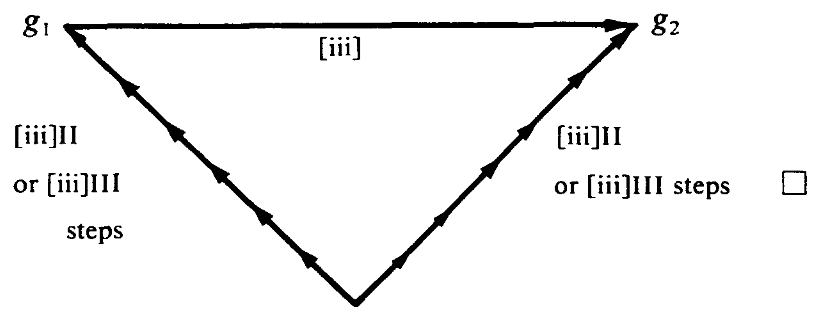
[iii]III



Proof. Every arc can be filled up with elementary arcs, e.g.:



Therefore, every arc reduction $g_1 \xrightarrow[\text{[iii]}]{} g_2$ can be replaced by a conversion consisting of elementary arc reductions of type [iii]II or [iii]III:



□

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 '*basic terms*'.

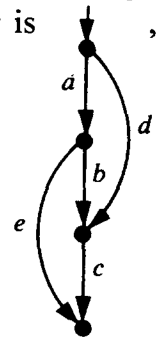
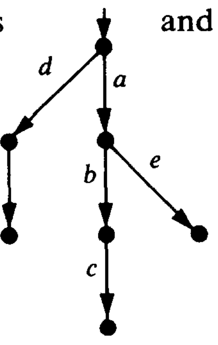
E.g., $a(b+c)d$ and $a(c+b)d$ are (representations of) the same basic term; $a(bd+cd)$ is (a representation of) a different basic term.

2.16. Definition. 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 $\mathcal{G}/\underline{\leftrightarrow}_{r\tau}$; so $[\![t]\!]$ is a process graph modulo $r\tau$-bisimulation.

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

Example. If g is , then g' is  and $\wr g \wr = dc + a(bc+e)$.

2.17. Proposition. *Let $g_1, g_2 \in G$ and suppose $g_1 \rightarrow g_2$ via an elementary graph reduction* [i], [ii], [iiiI, II, III], [iv], [v]. *Then the basic terms $\wr g_1 \wr$ and $\wr g_2 \wr$ can be proved equal using the* A*-axioms* (*about* $+, \cdot, \delta$) *in Table* 2, A1-7, *and the τ-laws* T1-3 (*see the diagram below*).

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

Proof. 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 Fig. 16(a)), 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 Fig. 16(b)). 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. □

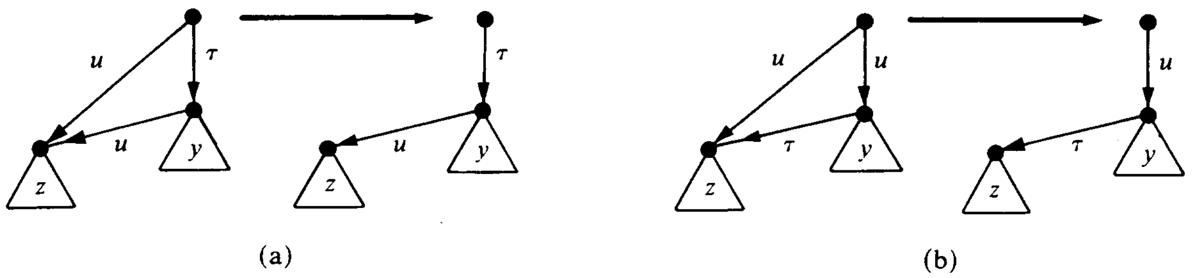


Fig. 16.

Now we can prove an important fact.

2.18. Lemma. *Suppose t, s are basic terms. Then*:

$$\mathcal{G}/\underline{\leftrightarrow}_{r\tau}\models t=s \;\Rightarrow\; \text{A1-7, T1-3}\vdash t=s.$$

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

$$[t]\equiv g_0 - g_1 - \cdots - g_n\equiv[s]. \quad (\text{Here} - \text{is} \rightarrow \text{or} \leftarrow.)$$

Now Proposition 2.17 states that

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

Since A1-7 $\vdash \wr[t]\wr=t$ 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 the following lemma.

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

$$\mathcal{G}/\underline{\leftrightarrow}_{r\tau}\models t=s \;\Rightarrow\; \text{A1-7}\vdash t=s.$$

2.20. Elimination Theorem. *Let t be a closed term in the signature of* ACP$_\tau$. *Then, using the axioms of* ACP$_\tau$ *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'.*

For the proof, see Appendix A.

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

2.21. Lemma

(i)

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

I.e., if $\mathrm{ACP}_\tau \vdash t_1 = t_2$, *then* t_1 *and* t_2 *can be reduced by means of the rewrite rules (from left to right) associated to the axioms in* $\mathrm{ACP}_\tau - \mathrm{AT}$ *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 $\mathrm{ACP}_\tau \vdash t_1 = t_2$. By the Elimination Theorem 2.20 we can rewrite t_1, t_2 to respectively basic terms t_3, t_4 using the axioms in $\mathrm{ACP}_\tau - \mathrm{AT}$ as rewrite rules. By the fact that $\mathcal{G}/\underline{\leftrightarrow}_{r\tau}$ is a model of ACP_τ we have $\mathcal{G}/\underline{\leftrightarrow}_{r\tau} \vDash t_3 = t_4$. Hence (Lemma 2.18) $\mathrm{AT} \vdash t_3 = t_4$.

(ii) Immediate from (i). □

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

(i)

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

(ii)

$$\begin{tikzcd}
a\tau \mathbin{\|\mkern-9mu\lfloor} b \arrow[rrrr, equal] \arrow[d] & & & & a \mathbin{\|\mkern-9mu\lfloor} b \arrow[ddd] \\
a(\tau \parallel b) \arrow[d] & & & & \\
a(\tau \mathbin{\|\mkern-9mu\lfloor} b + b \mathbin{\|\mkern-9mu\lfloor} \tau + \tau \mid b) \arrow[d] & & & & \\
a(\tau b + b\tau + \delta) & = a(\tau b + b\tau) & = a(\tau b + b) & = a\tau b & = ab
\end{tikzcd}$$

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

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

As a further corollary we have the following.

2.23. Theorem. (i) *$\mathcal{G}/\underline{\leftrightarrow}_{r\tau}$ is isomorphic to A_τ^ω, the initial algebra of* ACP$_\tau$.

(ii) ACP$_\tau$ *is conservative over* ACP (*the latter over the alphabet A*). *I.e., for τ-less terms t_1, t_2:*

$$\mathrm{ACP}_\tau \vdash t_1=t_2 \Rightarrow \mathrm{ACP} \vdash t_1=t_2.$$

Proof. (i) We have to prove

$$\mathcal{G}/\underline{\leftrightarrow}_{r\tau} \models s=t \Leftrightarrow \mathrm{ACP}_\tau \vdash s=t.$$

($\Leftarrow$) 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 $\mathrm{ACP}_\tau \vdash s=s'$, $t=t'$. 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 $\mathrm{ACP}_\tau \vdash t_1=t_2$. Let t_3, t_4 be basic terms such that $\mathrm{ACP}_\tau \vdash t_1=t_3$, $t_2=t_4$. Since t_3, t_4 can be obtained by rewrite rules $\mathrm{ACP}_\tau-\mathrm{AT}$, we have $\mathrm{ACP} \vdash t_1=t_3$, $t_2=t_4$. Now, by Lemma 2.19, A1-7 $\vdash t_3=t_4$. Hence, $\mathrm{ACP} \vdash t_1=t_2$. □

3. The Expansion Theorem for ACP$_\tau$

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 [12]. For ACP (i.e., ACP$_\tau$ without τ), the analogous theorem is proved in [5]. As an example we mention the Expansion Theorem for ACP in the case $k=3$:

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

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

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

(b) the *axioms of standard concurrency* for ACP (see Table 3).

Table 3.

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

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$_\tau$ this is no longer true; all axioms of standard concurrency hold in the initial algebra A_τ^ω of ACP$_\tau$ except the second one.

Example

$$(a \mid \tau b) \mathbin{\|\!\!\lfloor} c = (a \mid b)c \quad \text{and} \quad a \mid (\tau b \mathbin{\|\!\!\lfloor} c) = (a \mid b)c + (a \mid c)b + a \mid b \mid c.$$

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

Fortunately, the Expansion Theorem carries over from ACP to ACP$_\tau$ in exactly the same form. This is what we will prove in this section. The underlying intuition is that $\|$ and $\mathbin{\|\!\!\lfloor}$ behave in ACP$_\tau$ 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: $\tau x \to \tau x + x$. As an example, consider $\tau a \mid b$. Evaluated according to ACP, we have

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

However, according to ACP$_\tau$:

$$\tau a \mid b = a \mid b,$$

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,

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

just as in ACP$_\tau$.

Below, the proof of the Expansion Theorem will also entail the associativity of $\|$. Nevertheless, we have given in Appendix B 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$_\tau$ the first two τ-laws turn up frequently; this seems not to be the case for the third τ-law.)

3.1. Definition. T is the set of basic terms in normal form w.r.t. the rewrite rule associated to A4: $(x+y)z \to xz+yz$. (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 *tree*.)

3.2. Notation. 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.
Example. $a(\tau b + c) \sqsubseteq a(\tau b + c) + ab$.

3.3. Definition. Let $x \in T$. Then x is *saturated* if

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

Example. (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* $\mathrm{ACP}_\tau \vdash x = y$ (*in fact, even* $\mathrm{T2} \vdash x = y$).

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', noncommunicating 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 $\lfloor\!\lfloor$; for $|$ it is true under the condition that the arguments are saturated. Thus, we have the following proposition.

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) $\mathrm{ACP}_\tau \vdash x \,\|\, y = \tau_{\{t\}}(x^t \,\|\, y^t)$,
(ii) $\mathrm{ACP}_\tau \vdash x \lfloor\!\lfloor y = \tau_{\{t\}}(x^t \lfloor\!\lfloor y^t)$,
(iii) $\mathrm{ACP}_\tau \vdash \bar{x} \,|\, \bar{y} = \tau_{\{t\}}(\bar{x}^t \,|\, \bar{y}^t)$.

Proof. (i) Let

$$x = (\tau) + \sum a_i + \sum b_j x_j' + \sum \tau x_k'' \quad \text{and} \quad y = (\tau) + \sum c_l + \sum d_m y_m' + \sum \tau y_p''$$

where a_i, b_j, c_l, $d_m \in A$. Then

$$x \| y = x \mathbin{|\!\!\lfloor} y + y \mathbin{|\!\!\lfloor} x + x | y =$$

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

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

Claim. $x' \sqsubseteq x \;\&\; y' \sqsubseteq y \Rightarrow x' | y' \sqsubseteq x | y \sqsubseteq \tau(x \| y)$.

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

$$\mathrm{ACP}_\tau \vdash \tau(x \| y) = \tau(x \mathbin{|\!\!\lfloor} y + y \mathbin{|\!\!\lfloor} x + x | y) = \tau(x \mathbin{|\!\!\lfloor} y + y \mathbin{|\!\!\lfloor} x + x | y) + x | y.$$

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 give 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.:

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

follows by

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

and the induction hypothesis

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

(induction on the sum of the term complexities).

(ii) The case of $\mathbin{|\!\!\lfloor}$ is similar to that of $\|$.

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

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

where a_i, $b_j \in A$, $n, m, l \geqslant 0$, 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

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

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

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 TC4), $bx_1\,|\,\tau\bar{y}_2 = bx_1\,|\,\bar{y}_2$. 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:

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

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

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. (i) *Let* $I \subseteq A$ *be such that* $I\,|\,A = \{\delta\}$. (*Here* $I\,|\,A = \{c\,|\,\exists i \in I,\, a \in A,\, i\,|\,a = c\}$.) *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,*

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

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

$$\begin{aligned}
x_1 \| \ldots \| x_k &= \bar{x}_1 \| \ldots \| \bar{x}_k = \tau_{\{t\}}(\bar{x}_1^t \| \ldots \| \bar{x}_k^t) \\
&\underset{(*)}{=} \tau_{\{t\}}(\textstyle\sum \bar{x}_i^t \mathbin{\|\!\_} (\bar{X}_k^i)^t + \sum (\bar{x}_i^t | \bar{x}_j^t) \mathbin{\|\!\_} (\bar{X}_k^{i,j})^t) \\
&= \textstyle\sum \tau_{\{t\}}(\bar{x}_i^t \mathbin{\|\!\_} (\bar{X}_k^i)^t) + \sum \tau_{\{t\}}((\bar{x}_i^t | \bar{x}_j^t) \mathbin{\|\!\_} (\bar{X}_k^{i,j})^t) \\
&\underset{(**)}{=} \textstyle\sum (\tau_{\{t\}}(\bar{x}_i^t) \mathbin{\|\!\_} \tau_{\{t\}}(\bar{X}_k^i)^t) + \sum (\tau_{\{t\}}\bar{x}_i^t | \tau_{\{t\}}\bar{x}_j^t) \mathbin{\|\!\_} \tau_{\{t\}}(\bar{X}_k^{i,j})^t \\
&= \textstyle\sum \bar{x}_i \mathbin{\|\!\_} \bar{X}_k^i + \sum (\bar{x}_i | \bar{x}_j) \mathbin{\|\!\_} \bar{X}_k^{i,j} \\
&= \textstyle\sum x_i \mathbin{\|\!\_} X_k^i + \sum (x_i | x_j) \mathbin{\|\!\_} X_k^{i,j}.
\end{aligned}$$

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

Appendix A. Termination of ACP$_\tau$ reductions proved by recursive path orderings

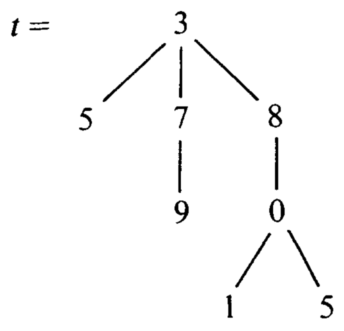
In this appendix we will prove the termination result in the Elimination Theorem 2.20 by the method of *recursive path orderings* as in [7]. Since we will give a slightly different presentation of recursive path orderings, a short account of this method will be given. Our presentation replaces Dershowitz's 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.

A.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.

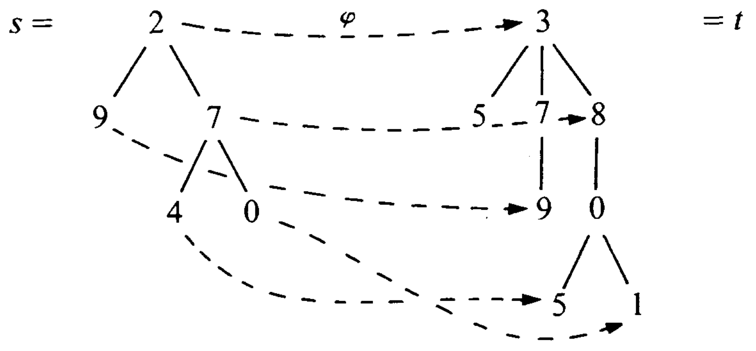
Example



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) $\rightarrow$ NODES(t) which is an order-preserving isomorphism and such that for all nodes $\alpha \in$ NODES(s) we have: label(α) $\leqslant$ label($\varphi(\alpha)$) where $\leqslant$ is the ordering on $\mathbb{N}$.

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



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

A.2. Kruskal Tree Theorem. *Let $t_1, t_2, t_3, \ldots$ be a sequence in D. Then for some $i<j$: $t_i \sqsubseteq t_j$.*

In fact, this is not the most general formulation of the theorem (see [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 [8].

The next definition is from [7].

A.3. Definition. The p.o. $\rhd$ on D is defined inductively as follows: $t=n(t_1,\ldots,t_k) \rhd m(s_1,\ldots,s_l)=s$ $(k, l \geqslant 0)$ iff

(i) $n > m$ and $t \rhd s_i$ for all $i = 1, \ldots, l$, or

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

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

It is implicit in [7] that an equivalent definition of $\rhd$ is the following.

A.4. Definition. The p.o. $\rhd$ on D is defined inductively as follows:

(a) $t = n(t_1, \ldots, t_k) \rhd m(s_1, \ldots, s_l) = s$ $(k, l \geq 0)$ iff

(i) as above, or

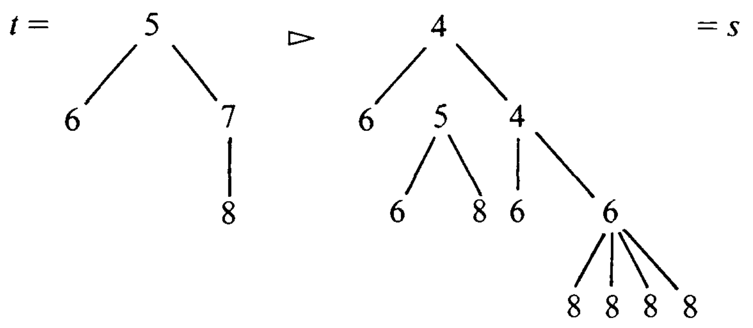
(ii) as above, or

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

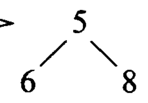
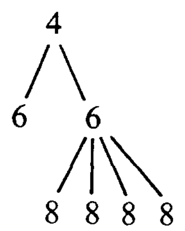
(b) $\rhd$ is transitive.

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

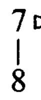
A.5. Example

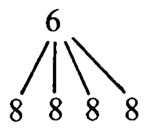


Proof. By (i) from Definition A.3, $t \rhd s$ if

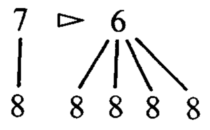
(a) $t \rhd 6$ and (b) $t \rhd$  and (c) t $\rhd$ 

(a) follows by (iii) of Definition A.3;

(b) follows by (ii) and  $\rhd 8$ (by (iii)).

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

(d) is by (iii) and (e) is so by (iii) since

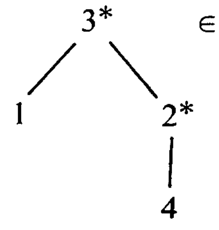


(by (i), (iii)). □

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.,

$3^*(1, 2^*(4)) =$



$\in D^*$.

Notation. If $t = n(t_1, \ldots, t_k)$ or $t = n^*(t_1, \ldots, t_k)$, then $t^* = n^*(t_1, \ldots, t_k)$.

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

A.6. Definition. On D^* a reduction relation $\Rightarrow$ is defined as follows:

(0) $n(t_1, \ldots, t_k) \Rightarrow n^*(t_1, \ldots, t_k)$ $(k \geq 0)$,

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

(2) $n^*(t_1, \ldots, t_k) \Rightarrow n(t_1^*, \ldots, t_1^*, t_2, \ldots, t_k)$ $(k \geq 1, s \geq 0$ copies of $t_1^*)$,

(3) $n^*(t_1, \ldots, t_k) \Rightarrow t_i$ $(i \in \{1, \ldots, k\}, k \geq 1)$,

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

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

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

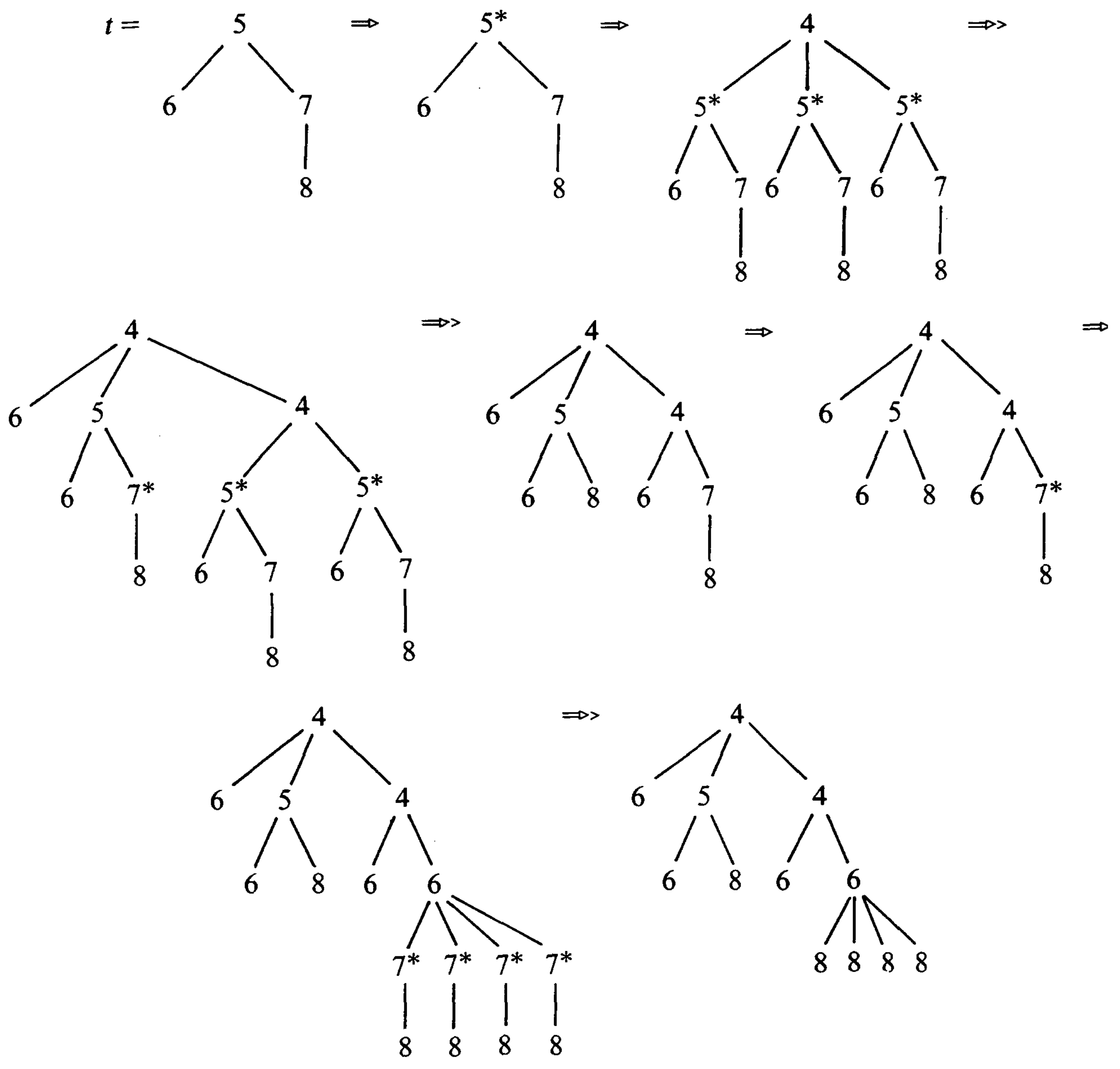
We are only interested in *-free $t \in D \subseteq D^*$. Now we have, by a tedious but routine proof which is omitted, the following proposition.

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

$$t \Rightarrow\!\!\!\Rightarrow s \ \textit{iff} \ t \unrhd s.$$

A.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 A.5):



In [7], the following facts about $\rhd$ are proved.

A.9. Proposition. $\rhd$ *is a partial order.*

The proof requires a simple induction to show the irreflexivity.

A.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 A.7, (i)-(iii) are immediate; e.g., (ii): $n(\vec{t}) \Rightarrow n^*(\vec{t}) \Rightarrow t_i$ and (i): $n(t_1, \ldots, t_k) \Rightarrow n^*(t_1, \ldots, t_k) \Rightarrow n(t_2, \ldots, t_k)$.

As to (iv): $n(\vec{t}) \Rightarrow n^*(\vec{t}) \Rightarrow m(n^*(\vec{t}), \ldots, n^*(\vec{t})) \Rightarrow\!\!\!> m(t_1, \ldots, t_k)$. □

Using Proposition A.10 one easily shows the following.

A.11. Proposition. $s \sqsubseteq t \implies t \trianglerighteq s$.

From this we have the following theorem.

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

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

A.13. Application to ACP$_\tau$. We want to prove that the rewrite rules (from left to right) associated to the axioms of ACP$_\tau$ except A1, 2, 5, C1, 2 and T1, 2, 3 are terminating. These rewrite rules have, in tree notation, the form as shown in Table 4 below.

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.

Therefore, we will rank the operators $\|, \lfloor\!\lfloor, |$ in the following way. Define the *weight* $|T|$ of a term T as follows:

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

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

$$|x+y| = \max\{|x|, |y|\},$$

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

Now the *rank* of an operator $\|, \lfloor\!\lfloor, |$ is the weight of the subterm of which it is the leading operator. Operators other than $\|, \lfloor\!\lfloor, |$ need no rank.

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

$$\|_n > \lfloor\!\lfloor_n, |_n \qquad \lfloor\!\lfloor_n, |_n > \|_{n-1} \qquad \|_n, \lfloor\!\lfloor_n, |_n > \cdot > + \qquad \partial_H, \tau_I > \cdot$$

(see Fig. 17).

Now consider a closed ACP$_\tau$-term T and obtain the ranked term T_r by assigning to all operators in T their rank.

Example

$$T = (ab \lfloor\!\lfloor cd) \lfloor\!\lfloor (\tau q \,|\, (r+uv))$$

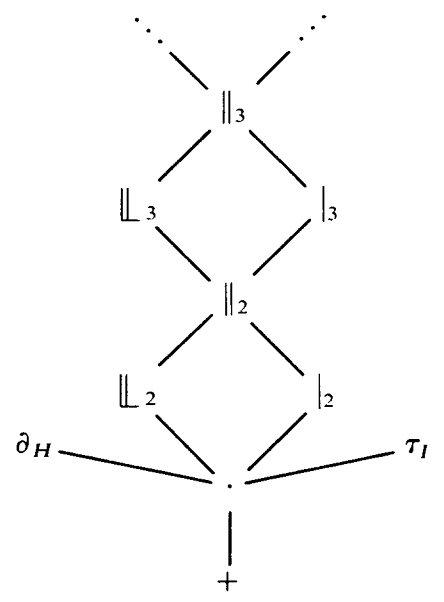
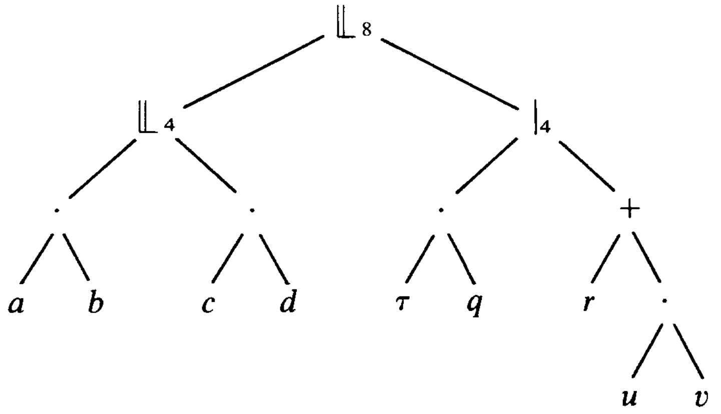


Fig. 17.

will be ranked as

$$T_r = (ab \lfloor\!\lfloor_4 cd) \lfloor\!\lfloor_8 (\tau q |_4 (r + uv)).$$

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



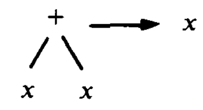
(In fact, we must assign to the a, τ, $\|_n$, $\lfloor\!\lfloor_n$, $|_n$, $+$, $\cdot$, ∂_H, τ_I natural numbers corresponding to the p.o. in Fig. 17 above. To all atoms we assign, say, 0.)

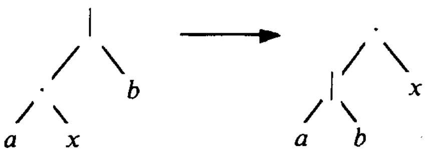
It is important to note that the definition of the rank of $\|, \lfloor\!\lfloor, |$ is such that a rewriting in term T does not cause the rank of operators 'higher' in the formation tree of T to increase (it may decrease). This is effectuated by the clause for $+$ in the definition of 'weight'. Indeed, if $T \to T'$ is a rewriting by one of the rewrite rules in Table 4, then $|T| \geq |T'|$, i.e., the weight cannot increase in a rewriting. Moreover, if $|T| \geq |T'|$, then for each context $C[\ ]$: $|C[T]| \geq |C[T']|$.

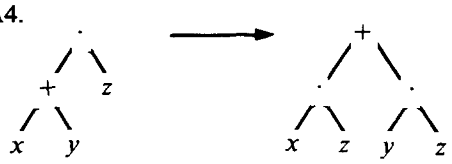
Now we have the following theorem.

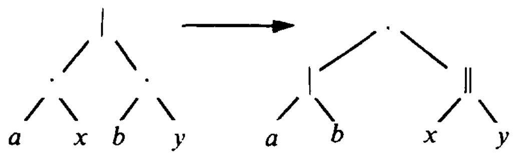
A.13.1. Theorem. *The rewrite rules in Table 4 have the termination property.*

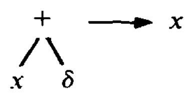
Table 4. Rewrite rules associated to the axioms of ACP_τ—{A1, 2, 5; C1, 2; T1, 2, 3}.

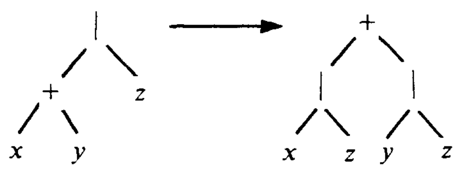
A3.


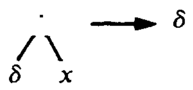
CM5, 6.


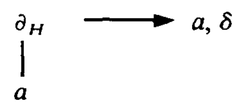
A4.


CM7.


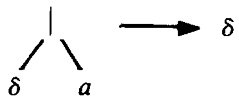
A6.


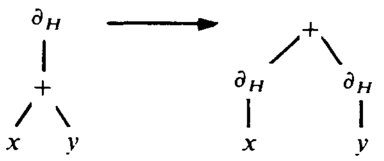
CM8, 9.


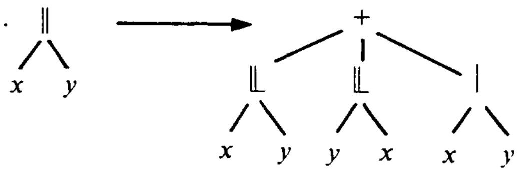
A7.


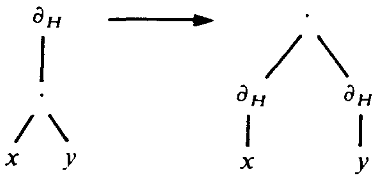
D1, 2.


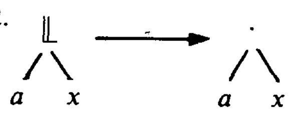
Likewise DT.

C3.


D3.


CM1.


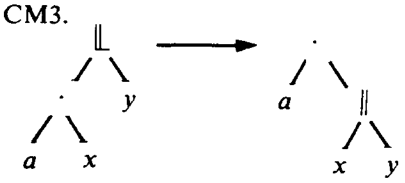
D4.


CM2.


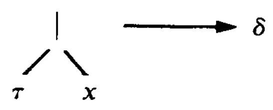
Likewise TM1.

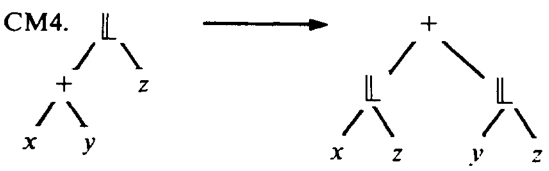
TI1-5:

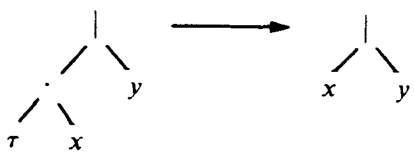
analogous to DT, D1-4.

CM3.


Likewise TM2.

TC1, 2.


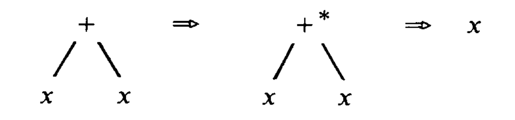
CM4.


TC3, 4.


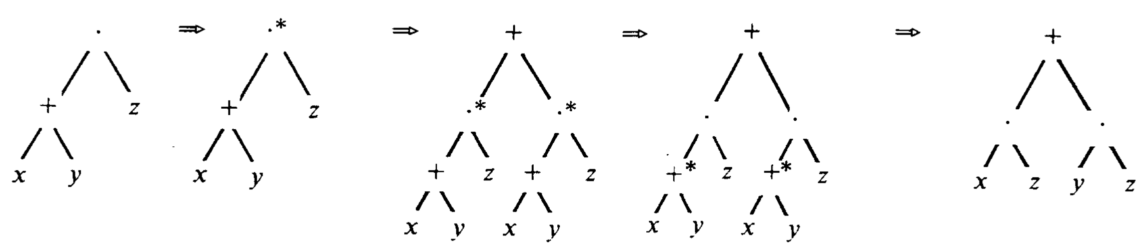
Proof. Let $\rhd$ be the recursive path ordering induced by the p.o. on the ranked operators as defined above. We will show that for each closed instance $t \to s$ of the rewrite rules, we have $t \rhd s$. In order to do so, we use the alternative definition of $\rhd$ as $\overset{+}{\Rightarrow}$ (the transitive closure of $\Rightarrow$). We will treat some typical cases (see Tables 5 and 6). □

Table 5.

A3.



A4.



CM1.

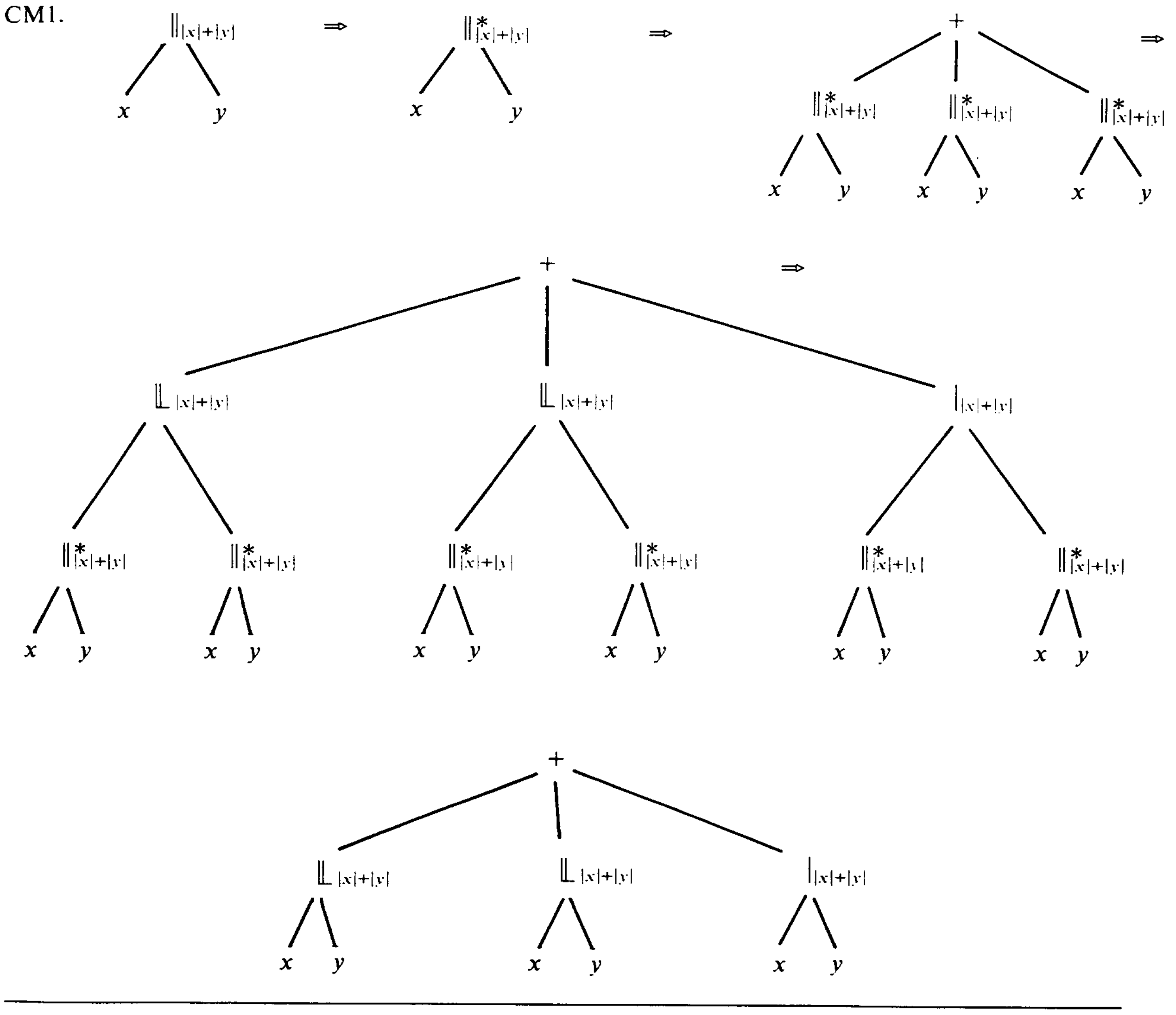
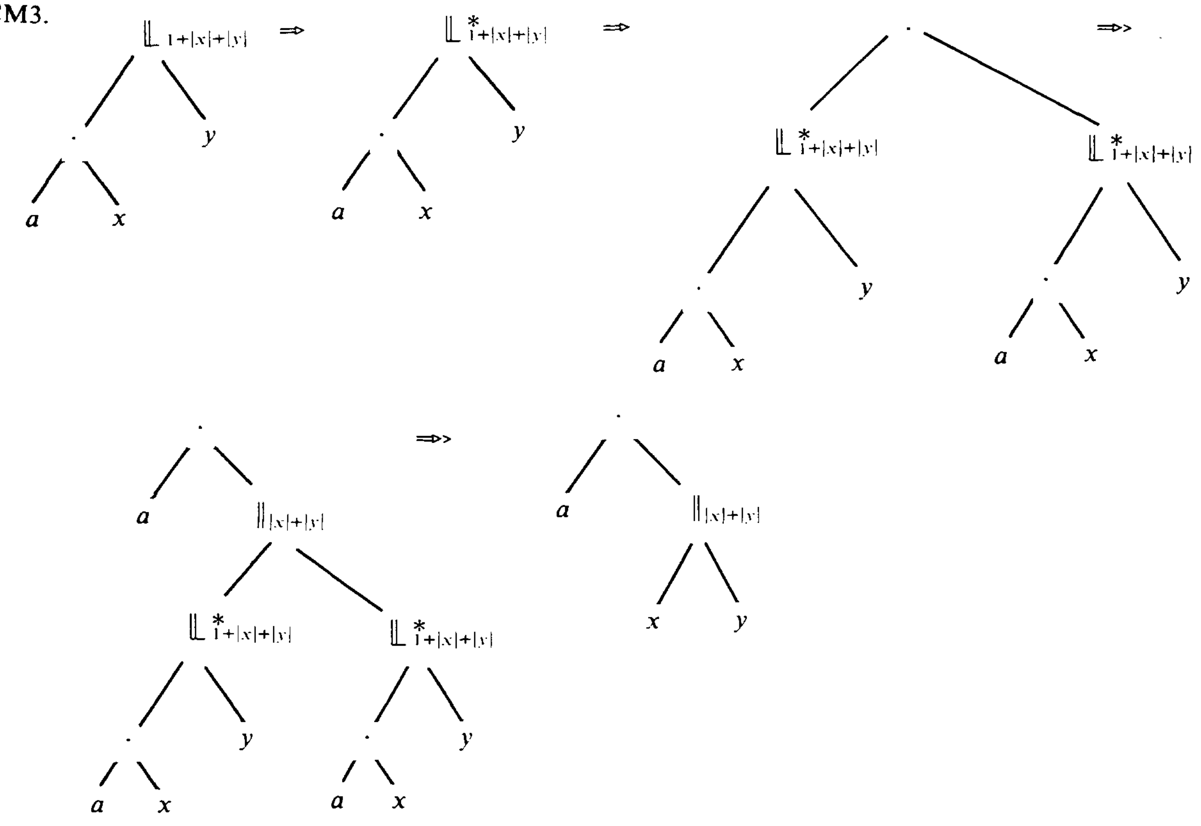
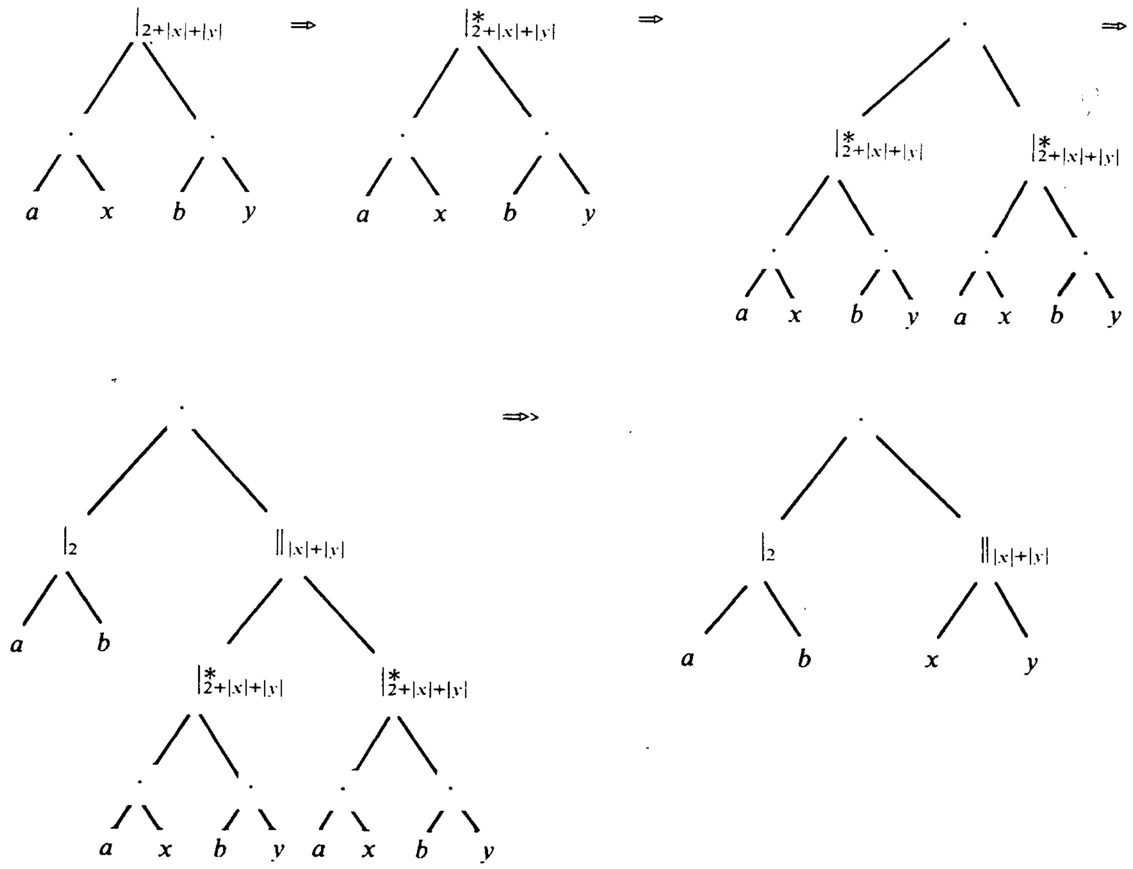


Table 6.

CM3.



CM7.



Appendix B. An inductive proof of associativity of merge in ACP$_\tau$

We will prove that in ACP$_\tau$ the identities as shown in Table 7 between closed terms are derivable. 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:

$$(x \mid y) \mathbin{\|\!\_}\, z = x \mid (y \mathbin{\|\!\_}\, z) \text{ if } y \text{ is stable.}$$

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

Table 7.

(1)	$(x \mathbin{\|\!\_}\, y) \mathbin{\|\!\_}\, z = x \mathbin{\|\!\_}\, (y \parallel z)$
(2)	$(x \mid ay) \mathbin{\|\!\_}\, z = x \mid (ay \mathbin{\|\!\_}\, z)$
(3)	$x \mid y = y \mid x$
(4)	$x \parallel y = y \parallel x$
(5)	$x \mid (y \mid z) = (x \mid y) \mid z$
(6)	$x \parallel (y \parallel z) = (x \parallel y) \parallel z$

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$_\tau$ 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$_\tau$-terms x, y, z.

B.1. Proposition. *Let $x, y, z \in T$. Then*:

(i) ACP$_\tau \vdash x \mid y = y \mid x$

(ii) ACP$_\tau \vdash x \parallel y = y \parallel x$.

Proof. Let $|x|$ be the length in symbols of x. The proof uses an induction on $|x| + |y|$. 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), $x \mid y = y \mid x$.

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

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

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

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) $x\|y=y\|x$:

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

B.2. Proposition. *Let* $x, y, z\in T$. *Then* $\mathrm{ACP}_\tau\vdash x|(y|z)=(x|y)|z$.

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.

B.3. Definition. Let x, y be closed ACP_τ-terms. Then we define: $\mathrm{ACP}_\tau\vdash x\sqsubseteq y$ if, for some closed term z, $\mathrm{ACP}_\tau\vdash y=x+z$.

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

B.4. Example
(i) $\mathrm{ACP}_\tau\vdash a\sqsubseteq\tau a$ (since $a=a+\tau a$),
(ii) $\mathrm{ACP}_\tau\vdash a\sqsubseteq a\|\tau$ (since $a\|\tau=\tau a+a\tau+a|\tau=\tau a+a$),
(iii) $\mathrm{ACP}_\tau\vdash\delta\sqsubseteq x$ for all x,
(iv) $\mathrm{ACP}_\tau\vdash a+\tau a+\tau b\overset{\sqsubseteq}{\sqsupseteq} b+\tau a+\tau b$.

B.5. Proposition. *Let* x, y *be closed terms. Then*:

$$\mathrm{ACP}_\tau\vdash x\sqsubseteq y\ \&\ \mathrm{ACP}_\tau\vdash y\sqsubseteq x\Rightarrow\mathrm{ACP}_\tau\vdash x=y.$$

Proof. We may suppose, by the Elimination Theorem 2.20, that $x, y\in T$. Suppose $\mathrm{ACP}_\tau\vdash y=x+z$ for some $z\in T$ and $\mathrm{ACP}_\tau\vdash x=y+u$ for some $u\in T$. Then $\mathrm{ACP}_\tau\vdash x=x+z+u$. 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 an $r\tau$-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 an $r\tau$-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 $\mathrm{ACP}_\tau \vdash x = x+z = y$. □

B.6. Proposition. *Let x be a closed term. Then* $\mathrm{ACP}_\tau \vdash x \mathbin{\|\!\!\_} \tau = x$.

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.

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

(i) $\mathrm{ACP}_\tau \vdash (x \mathbin{\|\!\!\_} y) \| z = x \mathbin{\|\!\!\_} (y \| z)$,

(ii) $\mathrm{ACP}_\tau \vdash (x | ay) \mathbin{\|\!\!\_} z = x | (ay \mathbin{\|\!\!\_} z)$,

(iii) $\mathrm{ACP}_\tau \vdash x \| (y \| z) = (x \| y) \| z$.

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): $(x \mathbin{\|\!\!\_} y) \| z = x \mathbin{\|\!\!\_} (y \| z)$.

Case (i)1. $x = x_1 + x_2$. Then $(x \mathbin{\|\!\!\_} y) \mathbin{\|\!\!\_} z = (x_1 \mathbin{\|\!\!\_} y) \mathbin{\|\!\!\_} z + (x_2 \mathbin{\|\!\!\_} y) \mathbin{\|\!\!\_} z =$ (induction hypothesis) $x_1 \mathbin{\|\!\!\_} (y \| z) + x_2 \mathbin{\|\!\!\_} (y \| z) = (x_1 + x_2) \mathbin{\|\!\!\_} (y \| z)$.

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

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

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

Next consider the induction step (ii): $(x | ay) \mathbin{\|\!\!\_} z = x | (ay \mathbin{\|\!\!\_} z)$. 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 $(x\,|\,ay) \mathbin{\lfloor\!\lfloor} z=(b\,|\,ay) \mathbin{\lfloor\!\lfloor} z=(b\,|\,a)y \mathbin{\lfloor\!\lfloor} z=(b\,|\,a)(y \parallel z)$, and also
$x\,|\,(ay \mathbin{\lfloor\!\lfloor} z)=b\,|\,(ay \mathbin{\lfloor\!\lfloor} z)=b\,|\,(a(y \parallel z))=(b\,|\,a)(y \parallel z)$.
Case (ii)5. $x=bx'$. Then $(x\,|\,ay) \mathbin{\lfloor\!\lfloor} z=(bx'\,|\,ay) \mathbin{\lfloor\!\lfloor} z=(b\,|\,a)(x' \parallel y) \mathbin{\lfloor\!\lfloor} z$
$=(b\,|\,a)((x' \parallel y) \parallel z)$, and $x\,|\,(ay \mathbin{\lfloor\!\lfloor} z)=bx'\,|\,(ay \mathbin{\lfloor\!\lfloor} z)=bx'\,|\,a(y \parallel z)$
$=(b\,|\,a)(x' \parallel (y \parallel z))$. 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): $L=x \parallel (y \parallel z)=(x \parallel y) \parallel z=R$. By the axioms in ACP_τ, we have

$$\begin{aligned}L=x \parallel (y \parallel z)&=x \mathbin{\lfloor\!\lfloor} (y \parallel z)+(y \parallel z) \mathbin{\lfloor\!\lfloor} x+x\,|\,(y \parallel z)\\&=x \mathbin{\lfloor\!\lfloor} (y \parallel z)+(y \mathbin{\lfloor\!\lfloor} z+y\,|\,z+z \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} x+x\,|\,(y \mathbin{\lfloor\!\lfloor} z+z \mathbin{\lfloor\!\lfloor} y+y\,|\,z)\\&=x \mathbin{\lfloor\!\lfloor} (y \parallel z)+(y \mathbin{\lfloor\!\lfloor} z) \mathbin{\lfloor\!\lfloor} x+(y\,|\,z) \mathbin{\lfloor\!\lfloor} x+(z \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} x+x\,|\,(y \mathbin{\lfloor\!\lfloor} z)\\&\quad+x\,|\,(z \mathbin{\lfloor\!\lfloor} y)+x\,|\,(y\,|\,z).\end{aligned}$$

Likewise, R can be expanded. We will use the following abbreviations: $L=l_1+\cdots+l_7$ and $R=r_1+\cdots+r_7$ where

$$\begin{aligned}&l_1=x \mathbin{\lfloor\!\lfloor} (y \parallel z), && r_1=(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z,\\&l_2=(y \mathbin{\lfloor\!\lfloor} z) \mathbin{\lfloor\!\lfloor} x, && r_2=(x\,|\,y) \mathbin{\lfloor\!\lfloor} z,\\&l_3=(y\,|\,z) \mathbin{\lfloor\!\lfloor} x, && r_3=(y \mathbin{\lfloor\!\lfloor} x) \mathbin{\lfloor\!\lfloor} z,\\&l_4=(z \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} x, && r_4=z \mathbin{\lfloor\!\lfloor} (x \parallel y),\\&l_5=x\,|\,(y \mathbin{\lfloor\!\lfloor} z), && r_5=(x \mathbin{\lfloor\!\lfloor} y)\,|\,z,\\&l_6=x\,|\,(z \mathbin{\lfloor\!\lfloor} y), && r_6=(y \mathbin{\lfloor\!\lfloor} x)\,|\,z,\\&l_7=x\,|\,(y\,|\,z), && r_7=(x\,|\,y)\,|\,z.\end{aligned}$$

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

From the Claim the induction step (iii) follows at once. Namely, we then have: $x \parallel (y \parallel z) \sqsubseteq (x \parallel y) \parallel z$, hence by Proposition B.1(ii): $x \parallel (y \parallel z) \sqsubseteq z \parallel (x \parallel y)$ (*). Now $z \parallel (x \parallel y)=z \parallel (y \parallel x) \sqsubseteq x \parallel (z \parallel y)=x \parallel (y \parallel z)$, where "$\sqsubseteq$" follows from (*). So we have $x \parallel (y \parallel z) \overset{\sqsubseteq}{\sqsupseteq} (x \parallel y) \parallel z$, and, by Proposition B.5: $x \parallel (y \parallel z)=(x \parallel y) \parallel z$.

The remainder of the proof is devoted to the proof of the above claim.

Proof of the Claim

(a) $l_7=r_7 \sqsubseteq R$ by Proposition B.2.

(b) $l_1 = r_1 \sqsubseteq R$ is statement (i) of this theorem; this induction step has already been proved. Likewise for $l_2 = r_3 \sqsubseteq R$ and $l_4 = r_4 \sqsubseteq R$.

(c) $l_3 \sqsubseteq r_6 \sqsubseteq R$. Here, $l_3 = (z\,|\,y) \mathbin{|\!\!\lfloor} x$ and $r_6 = z\,|\,(y \mathbin{|\!\!\lfloor} x)$.

Induction on z:

Case (iii)(c)1. $z = z_1 + z_2$. Then $l_3 = ((z_1+z_2)\,|\,y) \mathbin{|\!\!\lfloor} x = (z_1\,|\,y) \mathbin{|\!\!\lfloor} x$
$+ (z_2\,|\,y) \mathbin{|\!\!\lfloor} x \sqsubseteq$ (induction hypothesis) $z_1\,|\,(y \mathbin{|\!\!\lfloor} x) + z_2\,|\,(y \mathbin{|\!\!\lfloor} x)$
$= (z_1+z_2)\,|\,(y \mathbin{|\!\!\lfloor} x) = z\,|\,(y \mathbin{|\!\!\lfloor} x)$.

Case (iii)(c)2. $z = \tau$. Then $l_3 = r_6 = \delta$.

Case (iii)(c)3. $z = \tau z'$. Then $l_3 = (\tau z'\,|\,y) \mathbin{|\!\!\lfloor} x = (z'\,|\,y) \mathbin{|\!\!\lfloor} x \sqsubseteq z'\,|\,(y \mathbin{|\!\!\lfloor} x)$
$= \tau z'\,|\,(y \mathbin{|\!\!\lfloor} x) = z\,|\,(y \mathbin{|\!\!\lfloor} x)$.

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

Case (iii)(c)5. $z = az'$. To prove $(az'\,|\,y) \mathbin{|\!\!\lfloor} x \sqsubseteq az'\,|\,(y \mathbin{|\!\!\lfloor} x)$. We use an induction on y:

Case (iii)(c)5.1. $y = y_1 + y_2$. Then $(az'\,|\,(y_1+y_2)) \mathbin{|\!\!\lfloor} x = (az'\,|\,y_1) \mathbin{|\!\!\lfloor} x$
$+ (az'\,|\,y_2) \mathbin{|\!\!\lfloor} x \sqsubseteq az'\,|\,(y_1 \mathbin{|\!\!\lfloor} x) + az'\,|\,(y_2 \mathbin{|\!\!\lfloor} x)$
$= az'\,|\,((y_1+y_2) \mathbin{|\!\!\lfloor} x) = (az')\,|\,(y \mathbin{|\!\!\lfloor} x)$.

Case (iii)(c)5.2. $y = \tau$: $(az'\,|\,\tau) \mathbin{|\!\!\lfloor} x = \delta \mathbin{|\!\!\lfloor} x = \delta \sqsubseteq az'\,|\,(\tau \mathbin{|\!\!\lfloor} x)$.

Case (iii)(c)5.3. $y = \tau y'$: $(az'\,|\,\tau y') \mathbin{|\!\!\lfloor} x = (az'\,|\,y') \mathbin{|\!\!\lfloor} x \sqsubseteq (az')\,|\,(y' \mathbin{|\!\!\lfloor} x) \underset{(*)}{\sqsubseteq} (az')\,|\,(y' \parallel x)$
$\underset{(*)}{=} (az')\,|\,\tau(y' \parallel x) = (az')\,|\,(\tau y' \mathbin{|\!\!\lfloor} x)$. (Note the curious manoeuvre in steps (*).)

Case (iii)(c)5.4. $y = b$: $(az'\,|\,b) \mathbin{|\!\!\lfloor} x = ((a\,|\,b)z') \mathbin{|\!\!\lfloor} x = (a\,|\,b)(z' \parallel x) = (az')\,|\,(bx)$
$= (az')\,|\,(b \mathbin{|\!\!\lfloor} x)$.

Case (iii)(c)5.5. $y = by'$: $(az'\,|\,by') \mathbin{|\!\!\lfloor} x = ((a\,|\,b)(z' \parallel y')) \mathbin{|\!\!\lfloor} x = (a\,|\,b)(z' \parallel y') \parallel x)$
$= (a\,|\,b)(z' \parallel (y' \parallel x)) = (az')\,|\,b(y' \parallel x) = az'\,|\,((by') \mathbin{|\!\!\lfloor} x)$.

(d) Finally we prove $l_5 \sqsubseteq r_2 + r_5 + r_7 \sqsubseteq R$ (and by permuting x, y we then have also $l_6 \sqsubseteq r_2 + r_6 + r_7 \sqsubseteq R$), i.e.:

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

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 $x\,|\,(y \mathbin{|\!\!\lfloor} z) = x_1\,|\,(y \mathbin{|\!\!\lfloor} z) + x_2\,|\,(y \mathbin{|\!\!\lfloor} z)$
$\sqsubseteq (x_1\,|\,y) \mathbin{|\!\!\lfloor} z + (x_1 \mathbin{|\!\!\lfloor} y)\,|\,z + x_1\,|\,(y\,|\,z) + (x_2\,|\,y) \mathbin{|\!\!\lfloor} z$
$+ (x_2 \mathbin{|\!\!\lfloor} y)\,|\,z + x_2\,|\,(y\,|\,z) = (x\,|\,y) \mathbin{|\!\!\lfloor} z + (x \mathbin{|\!\!\lfloor} y)\,|\,z + x\,|\,(y\,|\,z)$.

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

Case (iii)(d)3. $x = \tau x'$. Then $\tau x'\,|\,(y \mathbin{|\!\!\lfloor} z) = x'\,|\,(y \mathbin{|\!\!\lfloor} z)$
$\sqsubseteq (x'\,|\,y) \mathbin{|\!\!\lfloor} z + (x' \mathbin{|\!\!\lfloor} y)\,|\,z + x'\,|\,(y\,|\,z)$
$= (\tau x'\,|\,y) \mathbin{|\!\!\lfloor} z + (x' \mathbin{|\!\!\lfloor} y)\,|\,z + \tau x'\,|\,(y\,|\,z)$
$\sqsubseteq (\tau x'\,|\,y) \mathbin{|\!\!\lfloor} z + (x' \parallel y)\,|\,z + \tau x'\,|\,(y\,|\,z)$
$= (\tau x'\,|\,y) \mathbin{|\!\!\lfloor} z + \tau(x' \parallel y)\,|\,z + \tau x'\,|\,(y\,|\,z)$
$= (\tau x'\,|\,y) \parallel z + (\tau x' \mathbin{|\!\!\lfloor} y)\,|\,z + \tau x'\,|\,(y\,|\,z)$
$= (x\,|\,y) \mathbin{|\!\!\lfloor} z + (x \mathbin{|\!\!\lfloor} y)\,|\,z + x\,|\,(y\,|\,z)$.

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

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

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

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

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

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

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

Case (iii)(d)5.4. Summands $\tau y''_l$ (for convenience we drop the subscript l and write $y=\tau y''+y^*$):
Now $ax'|(\tau y''\mathbin{\lfloor\!\lfloor} z)=ax'|\tau(y''\,\|\,z)=ax'|(y''\,\|\,z)$
$=ax'|(y''\mathbin{\lfloor\!\lfloor} z+z\mathbin{\lfloor\!\lfloor} y''+y'|z)$
$=ax'|(y''\mathbin{\lfloor\!\lfloor} z)+ax'|(z\mathbin{\lfloor\!\lfloor} y'')+ax'|(y''|z)\sqsubseteq$ (induction hypothesis)
$\sqsubseteq(ax'|y'')\mathbin{\lfloor\!\lfloor} z+(ax'\mathbin{\lfloor\!\lfloor} y'')|z+ax'|(y''|z)$
$+(ax'|z)\mathbin{\lfloor\!\lfloor} y''+(ax'\mathbin{\lfloor\!\lfloor} z)|y''+ax'|(y''|z)+ax'|(y''|z)$
$=$ (Here the first summand equals the fifth by (ii) of this theorem, and likewise the second equals the fourth.)
$=(ax'|y'')\mathbin{\lfloor\!\lfloor} z+(ax'\mathbin{\lfloor\!\lfloor} y'')|z+ax'|(y''|z)$
$=(ax'|y'')\mathbin{\lfloor\!\lfloor} z+(ax'\mathbin{\lfloor\!\lfloor} y'')|z+ax'|(\tau y''|z)$
$\sqsubseteq(ax'|y)\mathbin{\lfloor\!\lfloor} z+(ax'\mathbin{\lfloor\!\lfloor} y'')|z+ax'|(y|z)$.
This matches the RHS of $(*)$ except for the second summand. So it remains to prove:

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

Proof of $(**)$: induction on z.

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

Case (iii)(d)5.4.2. $z=\tau$: $(ax'\mathbin{\lfloor\!\lfloor} y'')|\tau=\delta\sqsubseteq\mathrm{RHS}(**)$.

Case (iii)(d)5.4.3. $z=\tau z'$: $(ax'\mathbin{\lfloor\!\lfloor} y'')|(\tau z')=(ax'\mathbin{\lfloor\!\lfloor} y'')|z'\sqsubseteq(ax'\mathbin{\lfloor\!\lfloor} y)|z'$
$=(ax'\mathbin{\lfloor\!\lfloor} y)|(\tau z')$.

Case (iii)(d)5.4.4. $z=b$: $(ax'\mathbin{\lfloor\!\lfloor} y'')|b=a(x'\,\|\,y'')|b=(a|b)(x'\,\|\,y'')$.
Now $x'\,\|\,y=x'\,\|\,(\tau y''+y^*)=x'\mathbin{\lfloor\!\lfloor}(\tau y''+y^*)+(\tau y''+y^*)\mathbin{\lfloor\!\lfloor} x'$
$+x'|(\tau y''+y^*)=\tau(y''\,\|\,x')+T$.
So: $(ax'\mathbin{\lfloor\!\lfloor} y)|b=(a|b)(x'\,\|\,y)=$
$(a|b)(\tau(y''\,\|\,x')+T)\underset{\uparrow}{=}(a|b)(\tau(y''\,\|\,x')+T)+(a|b)(y''\,\|\,x')$.
Here "$\underset{\uparrow}{=}$" is an application of the third τ-law, T3. Therefore,
$(ax'\mathbin{\lfloor\!\lfloor} y'')|b=(a|b)(x'\,\|\,y'')\sqsubseteq(ax'\mathbin{\lfloor\!\lfloor} y)|b$.

Case (iii)(d)5.4.5. $z=bz'$: similar.

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

References

[1] J.W. De Bakker and J.I. Zucker, Processes and the denotational semantics of concurrency, *Information and Control* **54** (1/2) (1982) 70-120.

[2] J.A. Bergstra and J.W. Klop, Process algebra for synchronous communication, *Information and Control* **60** (1-3) (1984) 109-137.

[3] J.A. Bergstra and J.W. Klop, An abstraction mechanism for process algebras, Report IW 231/83, Mathematisch Centrum, Amsterdam, 1983.

[4] J.A. Bergstra and J.W. Klop, Algebra of communicating processes, in: J.W. de Bakker, M. Hazewinkel and J.K. Lenstra, eds., *Proc. CWI Symp. Mathematics and Computer Science*, CWI Monograph Series (Centrum voor Wiskunde en Informatica, Amsterdam, 1985) to appear.

[5] J.A. Bergstra and J.V. Tucker, Top-down design and the algebra of communicating processes, *Sci. Comput. Program.* **5** (2) (1985) 171-199.

[6] S.D. Brookes and W.C. Rounds, Behavioural equivalence relations induced by programming logics, in: J. Diaz, ed., *Proc. 10th ICALP*, Barcelona, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983) 97-108.

[7] N. Dershowitz, Orderings for term-rewriting systems, *Theoret. Comput. Sci.* **17** (1982) 279-301.

[8] N. Dershowitz, A note on simplification orderings, *Inform. Process. Lett.* **9** (5) (1979) 212-215.

[9] S. Graf and J. Sifakis, A modal characterization of observational congruence on finite terms of CCS, in: J. Paredaens, ed., *Proc. 11th ICALP*, Antwerpen, Lecture Notes in Computer Science **172** (Springer, Berlin, 1984), 222-234

[10] M. Hennessy, A term model for synchronous processes, *Information and Control* **51** (1) (1981) 58-75.

[11] C.A.R. Hoare, A model for communicating sequential processes, in: R.M. McKeag and A.M. McNaughton, eds., *On the Construction of Programs* (Cambridge University Press, 1980) 229-243.

[12] R. Milner, *A Calculus for Communicating Systems*, Lecture Notes in Computer Science **92** (Springer, Berlin, 1980).

[13] G. Winskel, Synchronisation trees, in: J. Díaz, ed., *Proc. 10th ICALP*, Barcelona, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983) 695-711; also in: *Theoret. Comput. Sci.* **34** (1, 2) (1984) 33-82.