Two Finite Specifications of a Queue

Marc Bezem[1] Alban Ponse[2]

[1] *Utrecht University, Department of Philosophy*
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

[2] *University of Amsterdam, Programming Research Group*
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

E-mail: `bezem@phil.ruu.nl` `alban@fwi.uva.nl`

Abstract. Two finite specifications of a queue in ACP with abstraction are proven correct relative to a standard specification of a queue that employs an infinite data type for representing its contents. The proofs are given in the proof theory of μCRL, and the only 'τ-laws' used are $x\tau = x$ and $x(\tau(y+z)+y) = x(y+z)$. Therefore the proofs are adequate for both 'branching bisimilarity' and 'observation equivalence'. Additionally, it is shown that *standard concurrency* follows from RSP for a class of processes guardedly specifiable in ACP with abstraction.

Key words & Phrases: process algebra, queue, guardedness.
1987 CR Categories: F.1.2, F.3.1, F3.2.

1 Introduction

The purpose of this paper is to record correctness proofs of two finite specifications of a *queue*, introduced below. Both these specifications are already known for some time. However, of the first one no proof has been published yet (as far as we are aware), and only slight variants of the second one were proven correct. Furthermore, the specifications and proofs are given in the proof theory of μCRL [GP90, GP91, GP94], and can in that form be used as samples for proof checking.

Additionally, the paper offers a small theoretical result on RSP (Recursive Specification Principle), a fixed point rule mostly adopted in proofs on recursively specified processes in the setting of ACP with abstraction (see [BK85, BW90] for a general introduction to ACP with abstraction and RSP). This result states that RSP implies the *standard concurrency* identities for processes specified by linear equations (and therewith commutativity and associativity of $\|$). This is remarkable, because axioms for standard concurrency are often explicitly adopted in correctness proofs that already use RSP.

Queues and correctness. Proving that some specification indeed defines a queue presupposes a standard *definition* of a queue. Correctness then boils down to proving equality with this standard definition using specific axioms and rules. Given that a queue is a process that receives data of type D along some in-port, and can send data in the same order in which they were received along some out-port (FIFO-like, i.e., First In, First Out), we use sequences of D-elements, typed *Sequence*, to represent the contents of a queue at any moment in its execution. For a queue with in-port i and out-port j, this standard specification is given in μCRL by the following recursion equation (cf.

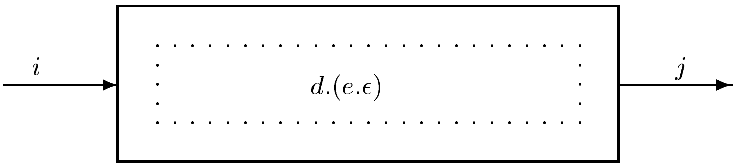
[BW90]):

$$\begin{aligned} Q^{ij}(q : \mathit{Sequence}) \quad &= \quad \textstyle\sum_{d\in D}(r_i(d)\cdot Q^{ij}(d.q)) + \\ & \quad\quad s_j(\mathit{right\text{-}elt}(q))\cdot Q^{ij}(\mathit{left\text{-}rm}(q)) \lhd \mathit{non\text{-}empty}(q) \rhd \delta \end{aligned}$$

where: . in $d.q$ is the function that inserts a new D element at the left of a sequence;
$P \lhd b \rhd Q$ abbreviates **if** b **then** P **else** Q **fi** (notation from [HHJ+87]);
$\mathit{non\text{-}empty}(q)$ is the Boolean expressing whether q does not equal ϵ, the *empty* sequence;
right-elt extracts the right element of a sequence, e.g. $\mathit{right\text{-}elt}(d.(e.\epsilon)) = e$;
left-rm extracts the remainder of a *Sequence* element, e.g. $\mathit{left\text{-}rm}(d.(e.\epsilon)) = d.\epsilon$.

The functions *left-rm* and *right-elt* are by definition total: $\mathit{left\text{-}rm}(\epsilon) = \epsilon$ and $\mathit{right\text{-}elt}(\epsilon) = d_0$ for some arbitrary $d_0 \in D$.

The parameter q in $Q^{ij}(q)$ represents the execution state of the queue. For example, one can depict $Q^{ij}(d.(e.\epsilon))$, a state that can be reached by having received first e and then d along port i (actions $r_i(e)$, $r_i(d)$), as follows:



Now $Q^{ij}(d.(e.\epsilon))$ can either receive some new datum e' along port i (action $r_i(e')$) and evolve into $Q^{ij}(e'.(d.(e.\epsilon)))$, or can send e along port j (action $s_j(e)$) and evolve into $Q^{ij}(d.\epsilon)$. In Section 3, we provide a detailed specification of the data type *Sequence* in μCRL style.

Starting point for this paper is to adopt the process $Q^{ij}(\epsilon)$ as the standard definition of the (empty) queue with in-port i and out-port j.

Two finite queue specifications in ACP with abstraction. Staying in the realm of ACP with abstraction, one *can* specify the queue above by using only the process operations of $\mathrm{ACP}_\tau(A,\gamma)$, instead of basing it on data type specification and data/process interaction. Two such specifications of a queue will be proven correct, i.e., equal to $Q^{ij}(\epsilon)$. The only 'τ-laws' used in this proof are

(B1) $x\tau = x$

(B2) $x(\tau(y+z)+y) = x(y+z)$

which were defined in the setting of 'branching bisimulation' [GW89, GW89a]. (These laws are implied by the τ-laws for 'observation equivalence' [Mil89].) In the following we introduce two finite queue specifications.

In the paper [BK86a], Bergstra and Klop presented the following intricate $\mathrm{ACP}_\tau(A,\gamma)$ specification of a queue using six recursion equations and one auxiliary (hidden) port:

$$Q^{ij} = \textstyle\sum_{d\in D}(r_i(d)\cdot(Q^{ik} \parallel_k s_j(d)\cdot Q^{kj})) \quad \text{for all } \{i,j,k\} = \{1,2,3\},$$

where: Q^{ij} represents a queue with in-port i and out-port j;
$x \parallel_k y$ is short for $\tau_{\{c_k(d)\mid d\in D\}} \circ \partial_{\{r_k(d),s_k(d)\mid d\in D\}}(x \parallel y)$;
$r_i(d)|s_i(d) = c_i(d)$ for $i \in \{1,2,3\}$ and $d \in D$ (and no other communications are defined).

Result 1.1. *In μCRL proof theory with* (B1) *and* (B2), $Q^{ij} = Q^{ij}(\epsilon)$ *for* $i,j \in \{1,2,3\}, i \neq j$.

In the same style as the specification of Q^{ij} above, one can specify a queue with in-port 1 and out-port j by using only *four* equations, employing an auxiliary port i and linked one element buffers. The one element buffer B^{ij} is a process that can receive some datum d along port i, after which it can only send d along port j, after which this behaviour is repeated. This specification is the following:

$$\left.\begin{array}{rcl} QB^{1j} & = & \sum_{d:D}(r_1(d) \cdot (QB^{1i} \parallel_i s_j(d) \cdot B^{ij})) \\ B^{ij} & = & \sum_{d:D}(r_i(d) \cdot s_j(d) \cdot B^{ij}) \end{array}\right\} \text{for all } \{i,j\} = \{2,3\},$$

with the communications and $\parallel_i$ as defined above.

In [Hoa85], Hoare already proved that queues can be specified as 'linked' one element buffers. Similar results were proved by Van Glabbeek and Vaandrager [GV89], and Brinksma and Kars [BK91]. Most close to pure $\mathrm{ACP}_\tau(A,\gamma)$ is the specification from [GV89] that employs chaining operators $\gg$ and $\ggg$.

Result 1.2. *In μCRL proof theory with* (B1) *and* (B2), $QB^{1j} = Q^{1j}(\epsilon)$ *for* $j \in \{2,3\}$.

Note on RSP. In the set-up of μCRL, i.e. processes that may interact with data, RSP applications are based on systems of equations that are guarded and that go with a substitution mechanism that allows data modification (see Appendix). In this respect, the μCRL version of RSP differs from the usual ACP version and all such applications are therefore displayed in a very detailed way.

Plan of this paper. In the next section we prove that the two finite queue specifications introduced above are guarded, by which each identifier Q^{ij}, QB^{1j} uniquely determines a process. In Section 3, a proof of Result 1.1 is provided and in Section 4, Result 1.2 is proved. In Section 5 attention is given to standard concurrency in the presence of RSP. An appendix on RSP in the setting of μCRL completes the paper.

Acknowledgement. Jos van Wamel is thanked for his careful proof reading.

2 Guardedness

Our first task is to show that each of the two specifications under consideration indeed specifies *some* process, i.e. has a unique solution for each identifier. This is only the case if these specifications are 'guarded'. For example, $X = a \cdot \tau_{\{a\}}(X)$ clearly does not specify a process: any process aP with P not containing a is a solution for this specification. There are various definitions for guardedness of recursive specifications. Here we use a liberal version, defined in two stages (taken from [BBP94b]). We employ the standard operational semantics of μCRL (see [GP90]), and speak of *steps* that can be performed by a declared, closed process expression.

Definition 2.1. Let P be an expression containing X. An occurrence of X in P is *τ-guarded* if P has a subexpression $a \cdot Q$, where $a \in A \cup \{\tau\}$ and Q contains this occurrence of X.
We call a recursive specification $E = \{X_j = T_j \mid j \in J\}$ *τ-guarded* if by substituting expressions T_j for occurrences X_j in the right-hand sides ('unfolding') a finite number of times, one can obtain the situation that every occurrence of every X_j in the unfolded right-hand sides is τ-guarded.
The specification E is *τ-founded* if none of the X_j gives rise to an infinite number of consecutive τ steps.
The specification E is *guarded* if it is both τ-guarded and τ-founded.

Though τ-foundedness can easily be rephrased in a more formal way, we refrain here from doing this and just give a few examples. [1] The earlier mentioned specification $X = a \cdot \tau_{\{a\}}(X)$ is τ-guarded, but not τ-founded: X can do an a step to $\tau_{\{a\}}(X)$, which can only perform an infinite number of consecutive τ steps. On the other hand, the specification $X = X$ is τ-founded, but not τ-guarded.

We recall the queue specification of Bergstra and Klop:

Definition 2.2. For all $\{i, j, k\} = \{1, 2, 3\}$ we define

$$Q^{ij} = \sum_{d \in D} r_i(d)(Q^{ik} \parallel_k s_j(d)Q^{kj})$$

where $x \parallel_k y$ is short for $\tau_{\{c_k(d)|d \in D\}} \circ \partial_{\{r_k(d), s_k(d)|d \in D\}}(x \parallel y)$.

We shall prove that these six defining equations form a guarded system, thus ensuring that the Q^{ij}'s are uniquely determined by RSP. Observe that the data parameter in $r_i(d)$ and $s_j(d)$ is irrelevant for this analysis. Therefore we shall omit it, as well as the $\sum$ operator. Accessible states of Q^{ij} will be parsed on their $\parallel_{\_}$ structure. To this end the notion of (i, j)-tree is defined.

Definition 2.3. For all $\{i, j, k\} = \{1, 2, 3\}$ we define by simultaneous induction:
1. Every Q^{ij} is an (i, j)-tree;
2. If T is an (i, j)-tree and T' an (j, k)-tree, then $\begin{array}{ccc} & \parallel_j & \\ \diagup & & \diagdown \\ T & & T' \end{array}$ is an (i, k)-tree;
3. If T is an (i, j)-tree, then $\begin{array}{ccc} & \parallel_j & \\ \diagup & & \diagdown \\ T & & s_k Q^{jk} \end{array}$ is an (i, k)-tree.

Obviously, leaves of an (i, j)-tree are either of the form $Q^{i'j'}$ or of the form $s_{j'}Q^{i'j'}$. The left-right ordering of an (i, j)-tree induces an ordering of its leaves. From now on we speak about leftmost, rightmost and adjacent leaves with respect to this ordering.

Lemma 2.4. *For every (i, j)-tree T the following holds:*
1. T can do a step r_i originating from its leftmost leaf, which is of the form $Q^{ij'}$ for some $j' \neq i$;
2. T can do a step s_j originating from its rightmost leaf if and only if this rightmost leaf is of the form $s_j Q^{i'j}$ for some $i' \neq j$;
3. T can do a step τ originating from two adjacent leaves of the form $s_{j'}Q^{i'j'}$ and $Q^{j'k'}$ $(i' \neq j',\ j' \neq k')$ if and only if T has two such adjacent leaves;
4. T can do no other steps than those enlisted above.

Proof. By simultaneous induction on the structure of the (i, j)-tree T.
Case $T \equiv Q^{ij}$: obvious.

Case $T \equiv \begin{array}{ccc} & \parallel_k & \\ \diagup & & \diagdown \\ T' & & T'' \end{array}$ with (i, k)-tree T' and (k, j)-tree T''. By the induction hypothesis the lemma holds for T' and T''.
As for 1: T can do the step r_i of T' since $i \neq k$.
As for 2: T can do a possible step s_j of T'' since $j \neq k$, and the condition is equivalent for T and T'' since they share their rightmost leaf.

[1] A precise definition of τ-foundedness can be found in [BBP94a]. In fact, the definition of guardedness can be further relaxed to τ-guardedness and *semi*-τ-foundedness, but we do not need this here. In [BG94] a related definition of guardedness is introduced.

As for 3: T can do all steps τ that T' and T'' can do, when the adjacent leaves are either both leaves of T', or of T''. Moreover, if the rightmost leaf of T' is of the form $s_k Q^{i'k}$, then, by the induction hypothesis 2, T' can do a step s_k. Also, by induction hypothesis 1, T'' can do a step r_k originating from its leftmost leaf. Now observe that the rightmost leaf of T' and the leftmost leaf of T'' are adjacent, so we are happy to see that s_k and r_k communicate into c_k, which results in a τ step of T due to the definition of $\|_k$.
As for 4: follows from the induction hypotheses $1-4$ for T' and T'' plus the observation that s_k and r_k above are encapsulated due to the definition of $\|_k$.

Case $T \equiv \begin{array}{c} \|_k \\ \diagup \quad \diagdown \\ T' \quad s_jQ^{kj} \end{array}$ with (i,k)-tree T': similar but simpler than the previous case (only τ steps from T'). ∎

Lemma 2.5. *If T is an (i,j)-tree, then all accessible states of T are (i,j)-trees.*

Proof. By induction on the number of steps, it suffices to verify that each of the steps $1-3$ from the previous lemma results in an (i,j)-tree.

As for 1: The leftmost leaf of T, which is of the form $Q^{ij'}$, is replaced by the (i,j')-tree $\begin{array}{c} \|_{k'} \\ \diagup \quad \diagdown \\ Q^{ik'} \quad s_{j'}Q^{k'j'} \end{array}$ with $\{i,j',k'\}=\{1,2,3\}$. The result is again an (i,j)-tree (to be proved by induction on T).
As for 2: The rightmost leaf of T, which is of the form $s_jQ^{i'j}$, is replaced by the (i',j)-tree $Q^{i'j}$. (Note that $s_jQ^{i'j}$ itself is not an (i',j)-tree, but this doesn't matter.) The result is again an (i,j)-tree (to be proved by induction on T).
As for 3: Combines 2 and 1 on adjacent leaves. The result is again an (i,j)-tree. ∎

Corollary 2.6. *All accessible states of Q^{ij} are (i,j)-trees.*

The converse of this corollary is not true:

$$\begin{array}{ccc} & \|_j & \\ \diagup & & \diagdown \\ Q^{ij} & & \|_i \\ & \diagup & \diagdown \\ & Q^{ji} & Q^{ik} \end{array} \quad \text{is an } (i,k)\text{-tree which is not accessible from } Q^{ik}.$$

Corollary 2.7. *The operational behaviour of an (i,j)-tree is completely determined by its list of leaves.*

Proof. By Lemma 2.5 and Lemma 2.4. The conditions on the clauses $1-3$ of Lemma 2.4 are expressed in terms of the leaves only. ∎

By the corollary above we can restrict attention to the leaves of an (i,j)-tree. We distinguish between Q-leaves of the form $Q^{i'j'}$ and Q_s-leaves of the form $s_{j'}Q^{i'j'}$. We represent sequences of leaves by sequences of natural numbers in the following way:

$$(n_1,...,n_k) \equiv Q^{n_1-1}Q_sQ^{n_2-1}Q_s...Q^{n_{k-1}-1}Q_sQ^{n_k},$$

where $k \geq 1$, $1 \leq i < k \to n_i \geq 1$, and Q^p represents a subsequence of p Q-leaves ($p \geq 0$) and Q_s represents a Q_s-leaf.

By inspection of the steps $1-3$ form Lemma 2.4 and the corresponding steps of the proof of Lemma 2.5 one gets the following transition system for (i,j)-trees:

$$
\begin{array}{rcll}
(n_1+1,n_2,...,n_k) & \xrightarrow{r_i} & (2,n_1,n_2,...,n_k) & \\
(n_1,..,n_{k-1},0) & \xrightarrow{s_j} & (n_1,...,n_{k-1}) & \text{provided } k>1 \\
(n_1,..,n_i+2,...,n_k) & \xrightarrow{\tau} & (n_1,...,n_{i-1}+2,n_i+1,...,n_k) & \text{provided } i>1 \\
(n_1,..,n_k+1) & \xrightarrow{\tau} & (n_1,...,n_{k-1}+2,n_k) & \text{provided } k>1.
\end{array}
$$

This transition system allows us to prove that the defining equations of Q^{ij} are τ-founded. Let $(n_1,...,n_k)$, with $k\geq 1$, represent such a state. Define:

$$
\begin{aligned}
&t_k = n_k \\
&t_i = n_i - 1 + 2t_{i+1} \qquad \text{for all } 1\leq i<k.
\end{aligned}
$$

Then we have $t_i=(n_i-1)2^0+(n_{i+1}-1)2^1+...+(n_{k-1}-1)2^{k-1-i}+n_k2^{k-i}$ for $1\leq i\leq k$. It is easily seen that for $k>1$ the maximum number of consecutive τ steps from the state $(n_1,...,n_k)$ is given by $t_2+...+t_k$, ending always in a state $(t_1+1,1,1,...,1,0)$. If $k=1$, then the state is (n_k) and there are no τ steps possible. In both states $(t_1+1,1,1,...,1,0)$ and (n_k) the process can only continue by an r_i or s_j step. We conclude that the defining equations of Q^{ij} are τ-founded.

Observe that the defining equations of Q^{ij} are trivially τ-guarded. Now we have the following theorem.

Theorem 2.8. *The defining equations of Q^{ij} are guarded.*

Remark 2.9. Theorem 2.8 justifies an application of RSP in an algebraic proof that Q^{ij} is a 'real' queue. This justification requires an analysis of the operational semantics of Q^{ij}. It is therefore questionable whether the algebraic proof is to be preferred over an argument showing that the operational semantics of Q^{ij} is weakly bisimilar to that of the 'real' queue. Such an argument can be obtained without much difficulty from the above analysis, by giving the atomic actions their data parameters back.

We now recall the queue specification inspired by [Hoa85]:

Definition 2.10. For all $\{i,j\}=\{2,3\}$ we define

$$
\begin{array}{rcl}
QB^{1j} & = & \sum_{d:D}(r_1(d)\cdot(QB^{1i}\parallel_i s_j(d)\cdot B^{ij})) \\
B^{ij} & = & \sum_{d:D}(r_i(d)\cdot s_j(d)\cdot B^{ij})
\end{array}
$$

where $x\parallel_k y$ is short for $\tau_{\{c_k(d)|d\in D\}}\circ\partial_{\{r_k(d),s_k(d)|d\in D\}}(x\parallel y)$.

In a similar way as Theorem 2.8 is proved, one can argue that this specification is guarded. Here one obtains a tree structure that is much simpler: QB^{1i} for $i\in\{2,3\}$ is a $(1,i)$-tree, and if T is a $(1,i)$-tree with leftmost leaf QB^{1k}, then replacing this leaf by the tree

$\begin{array}{c}\parallel_j\\ \diagup\quad\diagdown\\ QB^{1j}\quad s_kB^{jk}\end{array}$ or by the tree $\begin{array}{c}\parallel_j\\ \diagup\quad\diagdown\\ QB^{1j}\quad B^{jk}\end{array}$ (with $\{k,j\}=\{2,3\}$) again yields a $(1,i)$-tree. Without further proof we claim:

Theorem 2.11. *The defining equations of QB^{1i} are guarded.*

Table 1: The data type *Sequence* specified in μCRL.

sort	**Bool**	**var**	$d, e, f : D$
	D		$q, r : \mathit{Sequence}$
	Sequence		
		rew	$\mathit{right\text{-}elt}(\epsilon) = d_0$
func	$\mathbf{t}, \mathbf{f} :\rightarrow \mathbf{Bool}$		$\mathit{right\text{-}elt}(q.d) = d$
	$d_0 :\rightarrow D$		$\mathit{left\text{-}rm}(\epsilon) = \epsilon$
	$\vdots$		$\mathit{left\text{-}rm}(q.d) = q$
			$\mathit{non\text{-}empty}(\epsilon) = \mathbf{f}$
	$\epsilon :\rightarrow \mathit{Sequence}$		$\mathit{non\text{-}empty}(q.d) = \mathbf{t}$
	$. : \mathit{Sequence} \times D \rightarrow \mathit{Sequence}$		$d.\epsilon = \epsilon.d$
			$d.(q.e) = (d.q).e$
	$\mathit{right\text{-}elt} : \mathit{Sequence} \rightarrow D$		
	$\mathit{left\text{-}rm} : \mathit{Sequence} \rightarrow \mathit{Sequence}$		
	$\mathit{non\text{-}empty} : \mathit{Sequence} \rightarrow \mathbf{Bool}$		
	$. : D \times \mathit{Sequence} \rightarrow \mathit{Sequence}$		

3 The Bergstra-Klop Specification defines a Queue

In this section we prove that the specification of Bergstra and Klop discussed earlier indeed defines a queue (Result 1.1). We give this proof in an algebraic fashion, in the proof theory of μCRL.

The data type *Sequence*. In Table 1 we (partly) specify the data involved. The data type D, the elements of which are stored in the queue, is left unspecified apart from an arbitrary constant d_0. Notably D is not assumed to be finite.

The data type *Sequence* has two constructors, ϵ for the empty sequence, and a binary (infix written) function $. : \mathit{Sequence} \times D \rightarrow \mathit{Sequence}$ for inserting a D-element to the *right* of a sequence. Though this function is not used in the standard specification, it is convenient in our proofs. The function *right-elt* extracts the right element of a sequence, and gives the arbitrary element $d_0 \in D$ in case of emptyness, *left-rm* gives the left remainder of a sequence, and *non-empty* tests on inequality with ϵ. The standard queue specification employs the function $. : D \times \mathit{Sequence} \rightarrow \mathit{Sequence}$ modelling insertion of a D-element at the left side of a sequence, which is specified in the last two lines.

As a consequence of our choice of constructors, the induction principle for the data type *Sequence* reads as follows:

$$\text{(Induction Principle)} \quad \phi(\epsilon) \wedge \forall d \in D\ \forall q \in \mathit{Sequence}\ [\phi(q) \rightarrow \phi(q.d)]\ \rightarrow\ \forall q\ [\phi(q)].$$

Lemma 3.1. *Let $d : D$ and $q : \mathit{Sequence}$.*
1. $\mathit{non\text{-}empty}(q) = \mathbf{t} \rightarrow \mathit{right\text{-}elt}(d.q) = \mathit{right\text{-}elt}(q)$,
2. $\mathit{non\text{-}empty}(q) = \mathbf{t} \rightarrow \mathit{left\text{-}rm}(d.q) = d.\mathit{left\text{-}rm}(q)$, *and*
3. $\mathit{non\text{-}empty}(q) = \mathbf{t} \rightarrow \mathit{left\text{-}rm}(q).\mathit{right\text{-}elt}(q) = q$.

Proof. By induction. By way of example we give a proof of clause 3:
If $q = \epsilon$, then 3 follows trivially (from $\mathbf{f} = \mathbf{t}$ anything can be derived). In case $q = q'.d$, clause 3 follows also trivially from the equations defined in Table 1 and $\rightarrow$ introduction. ∎

Proving Result 1.1. Recall that for $i, j \in \{1,2,3\}, i \neq j$ our standard definition of a queue with in-port i and out-port j is $Q^{ij}(\epsilon)$ with specification

$$\begin{aligned} Q^{ij}(q : \mathit{Sequence}) &= \textstyle\sum_{d:D}(r_i(d) \cdot Q^{ij}(d.q)) + \\ &\quad s_j(\mathit{right\text{-}elt}(q)) \cdot Q^{ij}(\mathit{left\text{-}rm}(q)) \lhd \mathit{non\text{-}empty}(q) \rhd \delta. \end{aligned}$$

We prove that

$$Q^{ij}(\epsilon) = \sum_{d:D}(r_i(d) \cdot (Q^{ik}(\epsilon) \parallel_k s_j(d) \cdot Q^{kj}(\epsilon))) \quad \text{for all } \{i,j,k\} = \{1,2,3\}. \tag{1}$$

Because Q^{ij} (see Definition 2.2) has a guarded specification of exactly the same form (see Theorem 2.8), it follows by a trivial RSP application that

$$Q^{ij} = Q^{ij}(\epsilon).$$

Hence Q^{ij} also defines a queue with in-port i and out-port j. The rest of this section is devoted to this proof.

Identity (1). By symmetry, it suffices to prove (1) for $Q^{13}(\epsilon)$. For readability, assume for the remainder of this section that the following variables are globally declared:

$$\begin{aligned} &d, e, f : D \\ &q, r : \mathit{Sequence}. \end{aligned}$$

Define the auxiliary processes

$$\begin{aligned} X(q,r,d) &= \textstyle\sum_{e:D}(r_1(e) \cdot X(e.q,r,d)) + \\ &\quad s_3(d) \cdot Y(q,r) \\ Y(q,r) &= \textstyle\sum_{e:D}(r_1(e) \cdot Y(e.q,r)) + \\ &\quad s_3(\mathit{right\text{-}elt}(r)) \cdot Y(q, \mathit{left\text{-}rm}(r)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\ &\quad \tau \cdot Y(\mathit{left\text{-}rm}(q), \mathit{right\text{-}elt}(q).r) \lhd \mathit{non\text{-}empty}(q) \rhd \delta. \end{aligned}$$

Note that this specification is guarded: the summand starting with $\tau \cdot Y(...)$ does not give rise to an infinite τ trace, since the length of the first *Sequence* argument in $Y(...)$ decreases and the condition tests on emptyness.

The process $X(q,r,d)$ represents Q^{13} 'in state qrd' with its contents split up in a '$Q^{12}(q)$ part', a '$Q^{23}(r)$ part' and a '$s_3(d)$ part'. The process $X(q,r,d)$ can receive data and store these in its leftmost argument, or send the d in its rightmost argument and evolve into $Y(q',r)$. See also Figure 1.

The process $Y(q,r)$ is meant to represent $Q^{12}(q) \parallel_2 Q^{23}(r)$, and apart from performing r_1 and s_3 steps, it explicitly 'transfers' q to r by performing τ steps until q is ϵ. The process $Y(\epsilon,\epsilon)$ externally behaves as $Q^{13}(\epsilon)$. The process $Y(q,r)$ is depicted in Figure 2.

The following lemma, which plays an important role in our proof, states that these (internal) τ steps do not change the external behaviour of $\tau \cdot Y(q,r)$.

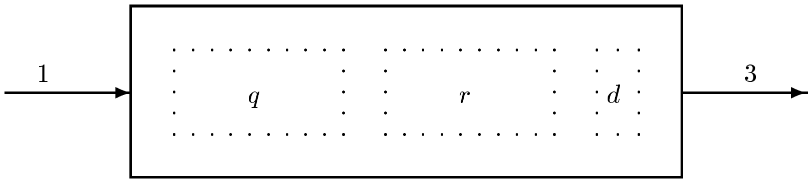


Figure 1: The auxiliary process $X(q, r, d)$.

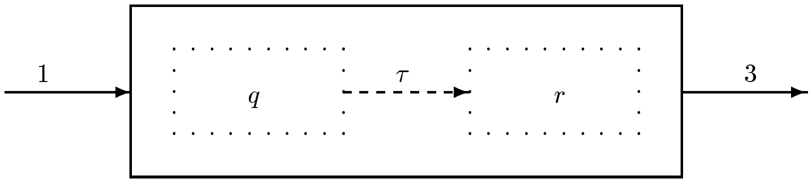


Figure 2: The auxiliary process $Y(q, r)$.

Lemma 3.2. *For all q, d, r it holds that $\tau \cdot Y(q.d, r) = \tau \cdot Y(q, d.r)$.*

Proof. By double expansion, $x \lhd \mathbf{t} \rhd y = x$, the last equation in Table 3.1 and Lemma 3.1,1 and 2, derive

$$
\begin{array}{lcll}
\tau \cdot Y(q.d, r) & = & \tau \cdot (& \sum_{e:D}(r_1(e) \cdot Y(e.(q.d), r)) + \\
 & & & s_3(\mathit{right\text{-}elt}(r)) \cdot Y(q.d, \mathit{left\text{-}rm}(r)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\
 & & & \tau \cdot Y(q, d.r) \;) \\
 & = & \tau \cdot (& \sum_{e:D}(r_1(e) \cdot Y((e.q).d, r)) + \\
 & & & s_3(\mathit{right\text{-}elt}(r)) \cdot Y(q.d, \mathit{left\text{-}rm}(r)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\
 & & & \tau \cdot \left[\begin{array}{l}
\sum_{e:D}(r_1(e) \cdot Y(e.q, d.r)) + \\
s_3(\mathit{right\text{-}elt}(r)) \cdot Y(q, d.\mathit{left\text{-}rm}(r)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\
s_3(d) \cdot Y(q, \epsilon) \lhd \mathit{not}(\mathit{non\text{-}empty}(r)) \rhd \delta + \\
\tau \cdot Y(\mathit{left\text{-}rm}(q), \mathit{right\text{-}elt}(q).(d.r)) \lhd \mathit{non\text{-}empty}(q) \rhd \delta
\end{array} \right]).
\end{array}
$$

Regarding d as the left element of r gives a similar identity, obtained with expansion and Lemma 3.1,1 and 2, followed by application of the following consequence of B2:

$$\tau \cdot (x + y + z + z') = \tau \cdot (x + y + \tau \cdot (x + y + z + z'))$$

(conversely, this identity implies B2, using BPA and B1).

$$\begin{array}{rcl}\tau\cdot Y(q,d.r) & = & \tau\cdot(\ \ \sum_{e:D}(r_1(e)\cdot Y(e.q,d.r))\ +\\ & & \quad s_3(\mathit{right\text{-}elt}(r))\cdot Y(q,d.\mathit{left\text{-}rm}(r))\lhd \mathit{non\text{-}empty}(r)\rhd\delta\ +\\ & & \quad s_3(d)\cdot Y(q,\epsilon)\lhd \mathit{not}(\mathit{non\text{-}empty}(r))\rhd\delta\ +\\ & & \quad \tau\cdot Y(\mathit{left\text{-}rm}(q),\mathit{right\text{-}elt}(q).(d.r))\lhd \mathit{non\text{-}empty}(q)\rhd\delta\ )\ )\\ & = & \tau\cdot(\ \ \sum_{e:D}(r_1(e)\cdot Y(e.q,d.r))\ +\\ & & \quad s_3(\mathit{right\text{-}elt}(r))\cdot Y(q,d.\mathit{left\text{-}rm}(r))\lhd \mathit{non\text{-}empty}(r)\rhd\delta\ +\\ & & \quad \tau\cdot\left[\begin{array}{l}\sum_{e:D}(r_1(e)\cdot Y(e.q,d.r))\ +\\ s_3(\mathit{right\text{-}elt}(r))\cdot Y(q,d.\mathit{left\text{-}rm}(r))\lhd \mathit{non\text{-}empty}(r)\rhd\delta\ +\\ s_3(d)\cdot Y(q,\epsilon)\lhd \mathit{not}(\mathit{non\text{-}empty}(r))\rhd\delta\ +\\ \tau\cdot Y(\mathit{left\text{-}rm}(q),\mathit{right\text{-}elt}(q).(d.r))\lhd \mathit{non\text{-}empty}(q)\rhd\delta\end{array}\right]).\end{array}$$

Now consider the guarded equation N_1 defined by

$$\begin{array}{rcl}n(q,d,r) & = & \tau\cdot(\ \ \sum_{e:D}(r_1(e)\cdot n(e.q,d,r))\ +\\ & & \quad s_3(\mathit{right\text{-}elt}(r))\cdot n(q,d,\mathit{left\text{-}rm}(r))\lhd \mathit{non\text{-}empty}(r)\rhd\delta\ +\\ & & \quad \tau\cdot\left[\begin{array}{l}\sum_{e:D}(r_1(e)\cdot Y(e.q,d.r))\ +\\ s_3(\mathit{right\text{-}elt}(r))\cdot Y(q,d.\mathit{left\text{-}rm}(r))\lhd \mathit{non\text{-}empty}(r)\rhd\delta\ +\\ s_3(d)\cdot Y(q,\epsilon)\lhd \mathit{not}(\mathit{non\text{-}empty}(r))\rhd\delta\ +\\ \tau\cdot Y(\mathit{left\text{-}rm}(q),\mathit{right\text{-}elt}(q).(d.r))\lhd \mathit{non\text{-}empty}(q)\rhd\delta\end{array}\right]).\end{array}$$

Applying τ-law B1 in the derivations above, it follows that both equations $N_1[\lambda qdr\ \ \tau\cdot Y(q.d,r)/n]$ and $N_1[\lambda qdr\ \ \tau\cdot Y(q,d.r)/n]$ are derivable. Hence RSP implies that $\tau\cdot Y(q.d,r)=\tau\cdot Y(q,d.r)$. ∎

The structure of the proof of Identity (1) is as follows:

$$\begin{array}{rcl}Q^{13}(\epsilon) & = & \sum_{d:D}(r_1(d)\cdot Q^{13}(d.\epsilon))\\ & \overset{(2)}{=} & \sum_{d:D}(r_1(d)\cdot Y(\epsilon,d.\epsilon))\\ & \overset{(3)}{=} & \sum_{d:D}(r_1(d)\cdot X(\epsilon,\epsilon,d))\\ & \overset{(4)}{=} & \sum_{d:D}(r_1(d)\cdot(Q^{12}(\epsilon)\parallel_2 s_3(d)\cdot Q^{23}(\epsilon)))\end{array}$$

showing which identities remain to be proved. The proofs given below all concern generalizations of these identities.

Identity (2). From Lemma 3.2 it follows that

$$\tau\cdot Y(d.\epsilon,q)=\tau\cdot Y(\epsilon.d,q)=\tau\cdot Y(\epsilon,d.q).$$

By expansion, B1 and the axioms for conditionals this leads to

$$\begin{array}{rcl}Y(\epsilon,q) & = & \sum_{d:D}(r_1(d)\cdot Y(d.\epsilon,q))+s_3(\mathit{right\text{-}elt}(q))\cdot Y(\epsilon,\mathit{left\text{-}rm}(q))\lhd \mathit{non\text{-}empty}(q)\rhd\delta\\ & = & \sum_{d:D}(r_1(d)\cdot Y(\epsilon,d.q))+s_3(\mathit{right\text{-}elt}(q))\cdot Y(\epsilon,\mathit{left\text{-}rm}(q))\lhd \mathit{non\text{-}empty}(q)\rhd\delta.\end{array}$$

By a trivial application of RSP we obtain:

$$Q^{13}(q)\ \ =\ \ Y(\epsilon,q). \tag{2}$$

Identity (3). First observe that

$$X(r.e, q, d) = X(r, e.q, d)$$

by RSP: consider the guarded equation N_2 defined by

$$n(r, e, q, d) \quad = \quad \sum_{f:D}(r_1(f) \cdot n(f.r, e, q, d)) + s_3(d) \cdot Y(r, e.q).$$

Now $N_2[\lambda reqd\ X(r.e, q, d)/n]$ is derivable (use Lemma 3.2), and so is $N_2[\lambda reqd\ X(r, e.q, d)/n]$ (immediate).

With the above observation, further derive

$$\begin{aligned} X(\epsilon, q, d) \quad &= \quad \sum_{e:D}(r_1(e) \cdot X(e.\epsilon, q, d)) + s_3(d) \cdot Y(\epsilon, q) \\ &= \quad \sum_{e:D}(r_1(e) \cdot X(\epsilon, e.q, d)) + s_3(d) \cdot Y(\epsilon, q). \end{aligned}$$

Because

$$\begin{aligned} Y(\epsilon, q.d) \quad &= \quad \sum_{e:D}(r_1(e) \cdot Y(e.\epsilon, q.d)) + s_3(d) \cdot Y(\epsilon, q) \\ &\overset{3.2}{=} \quad \sum_{e:D}(r_1(e) \cdot Y(\epsilon, e.(q.d))) + s_3(d) \cdot Y(\epsilon, q) \end{aligned}$$

we can apply RSP on the guarded equation M_2 defined by

$$n(q, d) \quad = \quad \sum_{e:D}(r_1(e) \cdot n(e.q, d)) + s_3(d) \cdot Y(\epsilon, q)$$

because both $M_2[\lambda qd\ X(\epsilon, q, d)/n]$ is derivable, and so is $M_2[\lambda qd\ Y(\epsilon, q.d)/n]$ (with the last rewrite rule in Table 1). It follows that

$$Y(\epsilon, q.d) \quad = \quad X(\epsilon, q, d). \tag{3}$$

Identity (4). First a trivial result on the commutativity of $\|$, namely $Q^{12}(q) \parallel Q^{23}(r) = Q^{23}(r) \parallel Q^{12}(q)$. Consider the guarded equation N_3 with only two occurrences of the identifier n:

$$\begin{aligned} n(q, r) \quad = \quad & \sum_{d:D}(r_1(d) \cdot (Q^{12}(d.q) \parallel Q^{23}(r))) + \\ & s_2(\mathit{right\text{-}elt}(q)) \cdot (Q^{12}(\mathit{left\text{-}rm}(q)) \parallel Q^{23}(r)) \lhd \mathit{non\text{-}empty}(q) \rhd \delta + \\ & \sum_{d:D}(r_2(d) \cdot (Q^{23}(d.r) \parallel Q^{12}(q))) + \\ & s_3(\mathit{right\text{-}elt}(r)) \cdot (Q^{23}(\mathit{left\text{-}rm}(r)) \parallel Q^{12}(q)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\ & c_2(\mathit{right\text{-}elt}(q)) \cdot n(\mathit{left\text{-}rm}(q), \mathit{right\text{-}elt}(q).r) \lhd \mathit{non\text{-}empty}(q) \rhd \delta. \end{aligned}$$

It follows easily that both $N_3[\lambda qr\ Q^{12}(q) \parallel Q^{23}(r)/n]$ and $N_3[\lambda qr\ Q^{23}(r) \parallel Q^{12}(q)/n]$ are derivable, whence this result follows with RSP. This trivial type of identities also follows from a general result on 'linearly specified' processes proven in Section 5.

With RSP and the identity above one proves

$$X(q, r, d) \quad = \quad Q^{12}(q) \parallel_2 s_3(d) \cdot Q^{23}(r) \tag{4}$$

$$Y(q, r) \quad = \quad Q^{12}(q) \parallel_2 Q^{23}(r) \tag{5}$$

simply by one expansion of all processes involved: for the system of guarded equations M_3 defined by

$$\begin{array}{rcl} n(q,r,d) & = & \sum_{e:D}(r_1(e)\cdot n(e.q,r,d)) + \\ & & s_3(d)\cdot m(q,r) \\ m(q,r) & = & \sum_{e:D}(r_1(e)\cdot m(e.q,r)) + \\ & & s_3(\mathit{right\text{-}elt}(r))\cdot m(q,\mathit{left\text{-}rm}(r)) \lhd \mathit{non\text{-}empty}(r) \rhd \delta + \\ & & \tau\cdot m(\mathit{left\text{-}rm}(q),\mathit{right\text{-}elt}(q).r) \lhd \mathit{non\text{-}empty}(q) \rhd \delta \end{array}$$

the derivability of

$$\begin{array}{l} M_3[\lambda qrd\ X(q,r,d)/n\ ,\ \lambda qr\ Y(q,r)/m] \quad \text{and} \\ M_3[\lambda qrd\ (Q^{12}(q)\parallel_2 s_3(d)\cdot Q^{23}(r))/n\ ,\ \lambda qr\ (Q^{12}(q)\parallel_2 Q^{23}(r))/m] \end{array}$$

follows immediately. This finishes the proof of Identity (1), and hence the proof of Result 1.1.

From the proof above it follows that two queues in parallel and connected via a hidden port externally behave as a single queue (cf. [BBP94a, Kor94]):

Corollary 3.3. *For all $\{i,j,k\}=\{1,2,3\}$ it holds that $Q^{ij}(\epsilon)=Q^{ik}(\epsilon)\parallel_k Q^{kj}(\epsilon)$.*

Proof. $Q^{ij}(\epsilon)\overset{(2)}{=}Y(\epsilon,\epsilon)\overset{(5)}{=}Q^{ik}(\epsilon)\parallel_k Q^{kj}(\epsilon)$. ∎

4 Queues as Linked One Element Buffers

In this section we prove for our standard queue specification that for all $\{i,j\}=\{2,3\}$

$$Q^{1i}(\epsilon)=\sum_{d:D}(r_i(d)\cdot(Q^{1j}(\epsilon)\parallel_j s_i(d)\cdot B^{ji})) \tag{6}$$

where B^{ij} is specified by the guarded equation $B^{ij}=\sum_{d:D}(r_i(d)\cdot s_j(d)\cdot B^{ij})$. By RSP it then follows immediately that QB^{1i} (see Definition 2.10, and for its guardedness Theorem 2.11) also defines a queue with in-port 1 and out-port i. This proves Result 1.2.

By symmetry, it suffices to prove (6) for $Q^{13}(\epsilon)$. The proof is in the same style as that of the previous result, and is given with less detail (though we maintain the claim on exactness).

Let d be a variable of type D and q be a variable of type *Sequence*. We define the following auxiliary processes:

$$\begin{array}{rcl} \overline{X}(q,d) & = & \sum_{e:D}(r_1(e)\cdot\overline{X}(e.q,d))+s_3(d)\cdot\overline{Y}(q) \\ \overline{Y}(q) & = & \sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q))+\tau\cdot\overline{X}(\mathit{left\text{-}rm}(q),\mathit{right\text{-}elt}(q)) \lhd \mathit{non\text{-}empty}(q) \rhd \delta. \end{array}$$

Both these are specified such that

$$\begin{array}{rcl} \overline{X}(q,d) & = & (Q^{12}(q)\parallel_2 s_3(d)\cdot B^{23}) \\ \overline{Y}(q) & = & (Q^{12}(q)\parallel_2 B^{23}) \end{array}$$

follow trivially (cf. identity (4) in the preceding proof). The structure of the proof of Identity (6) is as follows:

$$
\begin{array}{rcl}
Q^{13}(\epsilon) & = & \sum_{d:D}(r_1(d)\cdot Q^{13}(d.\epsilon)) \\
 & \overset{(8)}{=} & \sum_{d:D}(r_1(d)\cdot \overline{Y}(d.\epsilon)) \\
 & = & \sum_{d:D}(r_1(d)\cdot \overline{Y}(\epsilon.d)) \\
 & \overset{(7)}{=} & \sum_{d:D}(r_1(d)\cdot \overline{X}(\epsilon, d)) \\
 & = & \sum_{d:D}(r_1(d)\cdot (Q^{12}(\epsilon)\parallel_2 s_3(d)\cdot B^{23})).
\end{array}
$$

In the following, generalizations of the two marked identities are proved.

Identity (7). Consider the guarded equation M_4 defined by

$$
\begin{array}{rcl}
n(q,d) & = & \tau\cdot(\ \sum_{e:D}(r_1(e)\cdot n(e.q,d)) + \\
 & & \qquad \tau\cdot(\sum_{e:D}(r_1(e)\cdot \overline{X}(e.q,d)) + s_3(d)\cdot \overline{Y}(q)\ )\ ).
\end{array}
$$

From B1, B2 and the last equation in Table 3.1, it follows that both $M_4[\lambda qd\ \tau\cdot\overline{X}(q,d)/n]$ and $M_4[\lambda qd\ \tau\cdot\overline{Y}(q.d))/n]$ are derivable. Hence by RSP we find

$$\tau\cdot\overline{Y}(q.d) = \tau\cdot\overline{X}(q,d). \tag{7}$$

Identity (8). By expansion, B1 and Lemma 3.1.3 derive

$$
\begin{array}{rcl}
\tau\cdot\overline{Y}(q) & = & \tau\cdot(\sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q)) + \\
 & & \quad \tau\cdot\overline{X}(\textit{left-rm}(q),\textit{right-elt}(q)) \lhd \textit{non-empty}(q) \rhd \delta\ ) \\
 & = & \tau\cdot(\sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q)) + \\
 & & \quad \tau\cdot\left[\begin{array}{l}\sum_{e:D}(r_1(e)\cdot\overline{X}(e.\textit{left-rm}(q),\textit{right-elt}(q))) + \\ s_3(\textit{right-elt}(q))\cdot\overline{Y}(\textit{left-rm}(q))\end{array}\right] \lhd \textit{non-empty}(q) \rhd \delta\ ) \\
 & \overset{(7)}{=} & \tau\cdot(\sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q)) + \\
 & & \quad \tau\cdot\left[\begin{array}{l}\sum_{e:D}(r_1(e)\cdot\overline{Y}((e.\textit{left-rm}(q)).\textit{right-elt}(q))) + \\ s_3(\textit{right-elt}(q))\cdot\overline{Y}(\textit{left-rm}(q))\end{array}\right] \lhd \textit{non-empty}(q) \rhd \delta\ ) \\
 & \overset{(3.1.3)}{=} & \tau\cdot(\sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q)) + \\
 & & \quad \tau\cdot\left[\begin{array}{l}\sum_{e:D}(r_1(e)\cdot\overline{Y}(e.q)) + \\ s_3(\textit{right-elt}(q))\cdot\overline{Y}(\textit{left-rm}(q))\end{array}\right] \lhd \textit{non-empty}(q) \rhd \delta\ ).
\end{array}
$$

With B1 and B2 it follows easily that $\tau\cdot Q^{13}(q)$ has a similar expansion. [2] Hence we have by RSP that

$$\tau\cdot Q^{13}(q) = \tau\cdot\overline{Y}(q). \tag{8}$$

This finishes the proof of Identity (6), and hence the proof of Result 1.2.

[2]Derive the law $\tau\cdot(x+y\lhd b\rhd\delta) = \tau\cdot(x+\tau(x+y)\lhd b\rhd\delta)$ by case distinction on b.

Table 2: Axioms of Standard Concurrency (SC), with $a \in A$.

SC1	$(x \mathbin{⌊\!⌊} y) \mathbin{⌊\!⌊} z$	$=$	$x \mathbin{⌊\!⌊} (y \parallel z)$
SC2	$x \mathbin{⌊\!⌊} \delta$	$=$	$x\delta$
SC3	$x \mid y$	$=$	$y \mid x$
SC4	$(x \mid y) \mid z$	$=$	$x \mid (y \mid z)$
SC5	$x \mid (y \mathbin{⌊\!⌊} z)$	$=$	$(x \mid y) \mathbin{⌊\!⌊} z$
SC5-a	$x \mid (ay \mathbin{⌊\!⌊} z)$	$=$	$(x \mid ay) \mathbin{⌊\!⌊} z$

5 RSP and Standard Concurrency

Standard Concurrency (SC) is the axiomatic support for the identities

$$\begin{aligned} x \parallel y &= y \parallel x \\ (x \parallel y) \parallel z &= x \parallel (y \parallel z) \end{aligned}$$

expressing commutativity and associativity of $\parallel$ (see [BK85, BW90, GP91]). The axioms given in Table 2 and those of ACP allow one to derive these identities. All axioms in Table 2 are valid for closed (recursion-free) terms, to be proved by structural induction using the commutativity and associativity of the communication function on atomic actions.

The notion SC is a relative one. In the setting of observation congruence, the axiom SC5 is not sound: for example

$$a \mid (\tau b \mathbin{⌊\!⌊} c) \neq (a \mid \tau b) \mathbin{⌊\!⌊} c$$

because the left-hand term has a summand $(a|c)b$ which need not be equal to δ and which is absent in the right-hand term. Therefore, the alternative SC5-a is used in the setting of observation equivalence.

ACP and standard concurrency. Linear systems of recursion equations concern a syntactical characterization of certain processes. A system $\langle X_i \rangle_{i=1}^n$ of recursion equations is *linear* if it consists of equations of the form

$$X_i = \sum_{j=1}^{n} (\alpha_{ij} X_j) + \beta_i$$

for $i = 1, ..., n$, where α_{ij}, β_i are sums of atomic actions or equal δ. A process is *regular* iff it satisfies a (finite) linear system of recursion equations.

First observe that regular processes are closed under $\parallel, \mathbin{⌊\!⌊}, |, \partial_H$, and that this can be proved with RSP: expansion with linear systems as arguments always yields a linear system. RSP can also be used to prove commutativity and associativity of $\parallel$ for regular processes (and therewith the identities characterized by the standard concurrency axioms). This is because linearly specified processes expand recurrently in the scope of the parallel operator, giving way to RSP applications, as we show below.

Let for $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$ the regular processes X_i and Y_j be defined by

$$\begin{array}{rcl} X_i & = & \sum_{k=1}^{n}(\alpha_{ik}X_k) + \beta_i \\ Y_j & = & \sum_{l=1}^{m}(\overline{\alpha}_{jl}Y_l) + \overline{\beta}_j. \end{array}$$

The following guarded system, containing nm equations, serves to prove that $X_i \parallel Y_j = Y_j \parallel X_i$ for all appropriate i, j.

$$\begin{array}{rcl} n_{ij} & = & \sum_{k=1}^{n}(\alpha_{ik} \cdot (X_k \parallel Y_j)) + \sum_{l=1}^{m}(\overline{\alpha}_{jl} \cdot (Y_l \parallel X_i)) + \\ & & \beta_i Y_j + \overline{\beta}_j X_i + \\ & & \sum_{k=1}^{n}(\sum_{l=1}^{m}((\alpha_{ik}|\overline{\alpha}_{jl})n_{kl})) + \\ & & \sum_{k=1}^{n}((\alpha_{ik}|\overline{\beta}_j)X_k) + \sum_{l=1}^{m}((\overline{\alpha}_{jl}|\beta_i)Y_l) + (\beta_i|\overline{\beta}_j). \end{array}$$

Observe that both $X_i \parallel Y_j$ and $Y_j \parallel X_i$ are solutions for n_{ij}. This is a consequence of the commutativity of the communication function and the communication axioms CM 5,6 and CM 8,9. By RSP we conclude that $X_i \parallel Y_j = Y_j \parallel X_i$ for all appropriate i, j.

As to the associativity of $\parallel$, taking a third linear system $\langle Z_p \rangle_{p=1}^{r}$, the expansions of both $(X_i \parallel Y_j) \parallel Z_p$ and $X_i \parallel (Y_j \parallel Z_p)$ give identical patterns, apart from associative and commutative variants of the three identifiers and communications. It follows immediately that $\parallel$ also is an associative operation for regular processes.

Finally, given the commutativity and associativity of $\parallel$, the SC identities follow easily for linearly specified processes. As each recursion free process can be specified linearly, one obtains as a corollary that the SC axioms are valid for this class of processes.

This leads to the following result.

Theorem 5.1. *For the class of processes definable in* ACP *with linear recursive specifications,* RSP *implies that* $\parallel$ *is a commutative and associative operation, and that all axioms in Table 2 are valid.*

ACP with abstraction. In the setting with τ, linear systems may have τ as a constant. Adopting branching bisimilarity (i.e. the τ-laws B1 and B2), one can prove in the same way as above that RSP implies standard concurrency for all processes definable by guarded linear systems. In observation equivalence this requires more work, due to the additional τ-laws $\tau \cdot x \mid y = x \mid y$ and $x \mid \tau \cdot y = x \mid y$. In this case, a solution is to simultaneously prove the commutativity (associativity) of $\parallel$ and $\mid$.

Up to μCRL. It is not hard to see that the results of this section can be generalized to μCRL, though a general formulation of the appropriate type of specifications is cumbersome. By way of example, regard the proof of $Q^{12}(q) \parallel Q^{23}(r) = Q^{23}(r) \parallel Q^{12}(q)$ given in Section 3. Because the possible infinity of such systems is in this case captured by data, and because the possible τ occurrences do not affect the arguments above in a setting with data and conditionals, it follows immediately that the standard concurrency result is preserved in the μCRL setting. Finally, in [Pon92] it is shown that each recursive (finitely branching) transition system has a canonical 'μCRL-linear' specification modulo *strong bisimilarity*. Hence, *all* processes that can be associated with recursive transition systems satisfy the standard concurrency identities.

References

[BBP94a] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with combinators. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings CSL '93, Swansea*, volume 280 of *Lecture Notes in Computer Science*, pages 36–65. Springer-Verlag, 1994.

[BBP94b] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.

[BG94] M.A. Bezem and J.F. Groote. Invariants in process algebra with data. In B. Jonsson and J. Parrow, editors, *Proceedings CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*, pages 401–416. Springer-Verlag, 1994.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[BK86a] J.A. Bergstra and J.W. Klop. Process algebra: specification and verification in bisimulation semantics. In M. Hazewinkel, J.K. Lenstra, and L.G.L.T. Meertens, editors, *Mathematics and Computer Science II*, CWI Monograph 4, pages 61–94. North-Holland, Amsterdam, 1986.

[BK91] E. Brinksma and P. Kars. From data structure to process structure. Report MI 91-38 TIOS 91-11, University of Twente, Enschede, 1991.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[GP90] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. Report CS-R9076, CWI, Amsterdam, 1990. To appear in A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes, Utrecht 1994*, Workshops in Computing, Springer-Verlag, 1994.

[GP91] J.F. Groote and A. Ponse. Proof theory for μCRL. Report CS-R9138, CWI, Amsterdam, 1991.

[GP94] J.F. Groote and A. Ponse. Proof theory for μCRL: a language for processes with data. In D.J. Andrews, J.F. Groote, and C.A. Middelburg, editors, *Proceedings of the International Workshop on Semantics of Specification Languages*, pages 232–251. Workshops in Computing, Springer-Verlag, 1994.

[GV89] R.J. van Glabbeek and F.W. Vaandrager. Modular specifications in process algebra – with curious queues (extended abstract). In M. Wirsing and J.A. Bergstra, editors, *Algebraic Methods: Theory, Tools and Applications, Workshop Passau 1987*, volume 394 of *Lecture Notes in Computer Science*, pages 465–506. Springer-Verlag, 1989.

[GW89] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.

[GW89a] R. J. van Glabbeek and W. P. Weijland. Refinement in branching time semantics. Report CS-R8922, CWI, Amsterdam, 1989. Also appeared in: *Proceedings AMAST Conference*, May 1989, Iowa, USA, pp. 197–201.

[HHJ+87] C.A.R. Hoare, I.J. Hayes, He Jifeng, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey, and B.A. Sufrin. Laws of programming. *Communications of the ACM*, 30(8):672–686, August 1987.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, Englewood Cliffs, 1985.

[Kor94] H. Korver. *Protocol Verification in μCRL.* PhD thesis, University of Amsterdam, 1994.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[Pon92] A. Ponse. Computable processes and bisimulation equivalence. Report CS-R9207, CWI, Amsterdam, January 1992.

Appendix: RSP in μCRL

In this appendix we discuss the use of RSP in μCRL. In order to derive identities between μCRL-processes, an extended version of the Recursive Specification Principle (RSP, see eg. [BW90]) has been introduced in [GP91, GP94]. This was done to handle data parametric, conditional processes. Given a μCRL specification E, the (extended) rule RSP employs a system of process equations that is 'fresh' with respect to E and that must be 'guarded'.

Let a μCRL specification E be given and let $n_1, \ldots, n_m$ be m different fresh process identifiers. We call a system G of m equations $G_1, \ldots, G_m$ a system of *process-equations over E* iff

- each equation G_i has at its left-hand side an expression of the form n_i or $n_i(x_{i1}, \ldots, x_{im_i})$ where each x_{ij} is a data variable over data declared in E, and
- the right hand side of each equation G_i is a (well-formed) process expression over E that may contain the new, properly typed identifiers n_i or $n_i(t_{i1}, \ldots, t_{im_i})$ for $1 \leq i \leq m$.

The rule RSP in the setting of μCRL is restricted to systems of process-equations that are *guarded.* Though various characterizations of guardedness occur in the ACP/μCRL literature [BW90, GP90, GP91, BG94, BBP94b], we here stick to Definition 2.1 and add the following comments:

1. The guardedness of G refers to the specification of all process identifiers occurring in G;
2. τ-guardedness is a special case of the definition of guardedness in [GP91, GP94];
3. τ-foundedness can be rephrased in a formal way using the standard operational semantics for μCRL defined in [GP90].

Next we recall the substitution mechanism for a system $G = G_1, \ldots, G_m$ of process-equations over E defined in [GP91, GP94]. Abbreviating the (possible) variables of n_i by $\bar{x}_i$,

$$G_i[\lambda\bar{x}_i \;\, p(\bar{x}_i)/n_i]$$

is defined as the equation obtained by substituting $\lambda\bar{x}_i \;\, p(\bar{x}_i)$ for the n_i-occurrences in G_i, and then repeatedly performing β-conversion on the respective arguments of the process identifier n_i. For any identifier without arguments only the substitution of p is performed.

Given a guarded system $G_1, \ldots, G_m$ of m process-equations over E, the μCRL version of the rule RSP is as follows:

$$\frac{G_i[\lambda\bar{x}_j \;\, p_j(\bar{x}_j)/n_j]_{j=1}^{m} \qquad G_i[\lambda\bar{x}_j \;\, q_j(\bar{x}_j)/n_j]_{j=1}^{m}}{p_k(\bar{x}_k) = q_k(\bar{x}_k) \quad (1 \leq k \leq m)} \quad 1 \leq i \leq m$$

where

- for $1 \leq i \leq m$ the $p_i(\bar{x}_i)$ and $q_i(\bar{x}_i)$ are well-formed process terms over E,
- the notation $[\ldots]_{j=1}^{m}$ abbreviates the m given, simultaneous substitutions.

Further information on μCRL syntax, semantics and proof theory can be found in [GP90, GP91, BG94, GP94].