Process Algebra as a Tool for the Specification and Verification of CIM-architectures

S. Mauw
Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

Flexibility of a manufacturing system implies that it must be possible to reorganize the configuration of the system's components efficiently and correctly. To avoid costly redesign, we have the need for a formal description technique for specifying the (co)operation of the components. Process algebra - a theory for concurrency - will be shown to be expressive enough to specify, and even verify, the correct functioning of such a system. This will be demonstrated by formally specifying and verifying two workcells, which can be viewed as units of a small number of cooperating machines.

1. Introduction

One can speak of *Computer Integrated Manufacturing* (CIM) if the computer is used in all phases of the production of some industrial product. In this paper we will focus on the design of the product-flow and the information-flow, which occurs when products are actually produced. Topics like product-development, marketing and management are beyond the scope of this paper. The technique used in this paper is based on a theory for concurrency, called *process algebra* (see [4] or [5]). It can be used to describe the total phase of manufacturing, from the ordering of raw materials up to the shipping of the products which are made from this materials. During this process many machines are used, which can operate independently, but often depend on the correct operation of each other. Providing a correct functioning of the total of all machines, computers and transport-services is not a trivial exercise. Before actually building such a system (a *CIM-architecture*) there must be some design. Such a specification, when validated, describes a properly functioning system. The current trend towards *Flexible Manufacturing Systems* (FMS) introduces the need for a tool, able to validate a new design of a plant, before implementing it. The possibilities to use methods developed in process algebra for *specification* and *verification* of concurrent systems are described in this paper.

From a high level of view, a plant can be seen as constructed from several

Partial support received from the European Community under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

concurrently operating *workcells* (W1-W5 in Figure 1). Every workcell is responsible for some well-defined part of the manufacturing process, e.g. filling and capping a number of milk bottles. The various workcells are connected to each other via some transport-service, which manages input and output of goods for the workcells (the *logistics*). Of course some supervisor (control) must keep track of the (co)operation of all workcells. This control has connections to all other components of the plant, along which commands and status-reports are transmitted. The components labeled *supply* and *shipping* are used to store raw materials and processed goods.

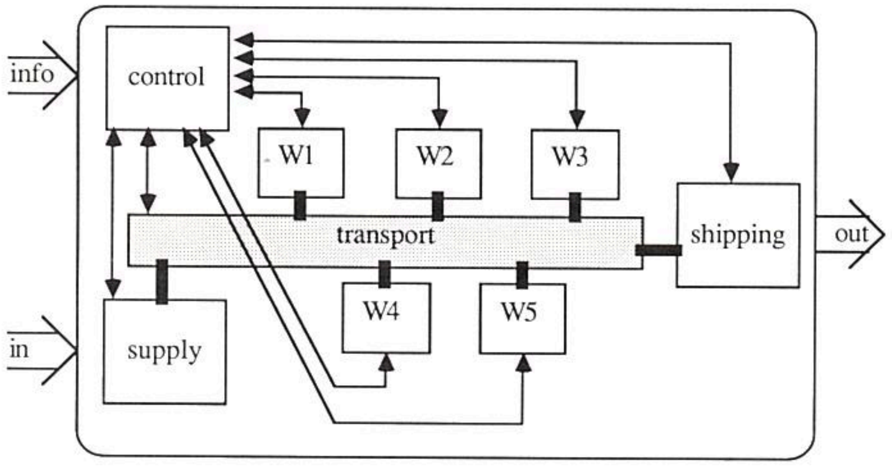

FIGURE 1. A sample architecture of a plant

Seen from a lower level, each workcell is constructed from a number of basic components which can perform one function, e.g. drilling a hole or assembling two parts. For controlling the communication with the outside and to instruct the various components of the workcell, each workcell has a *workcell-controller*. Also some simple transport-system must be present to transport the products within the workcell (see Figure 2).

The description of the components of some workcell can be given using process algebra. When abstracting from the internal actions of that workcell, it is possible to determine its external behaviour. At the high level view on the flow of products, we are only interested in the products which enter the workcell and the products leaving it. Also at the high level view on the flow of information, we only look at the commands we give a workcell to produce or process a number of products and the status-reports sent back.

The simple two level view on a manufacturing process expressed above, can be refined into a multi layered model, as is done in e.g. [7].

As an illustration of the technique we specify and verify two workcells in the theory ACP_τ (see [5]). The first one is a very simple one, able to produce and process one kind of product. The second one is more involved. It has the possibility to process some input product either correctly or faultily. Part of the workcell is a quality-check tool, which decides upon rejecting the product or not.

One should notice that in process algebra no real-time aspects are captured. So the important notions of *efficiency* (maximal productivity of the machines) and *tuning* (synchronization of the speed of the machines) cannot be modeled.

NOTE. This paper is partially based on discussions with F. Biemans, and inspired by his article [8], who used the specification language LOTOS (see [9]) to describe CIM-architectures. Other applications of theories for concurrency to CIM can be found in [10] and [11].

2. A SIMPLE WORKCELL

2.1. *Specification*

2.1.1. General description. In this section a simple workcell will be specified and verified, which consists of four components (see Figure 2). This workcell is identical to the one described in Biemans and Blonk [8]. Workstation A (*WA*) produces a product (product1) and offers this to the Transport service (*T*). Then the product is transported to Workstation B (*WB*), which processes the product and outputs it to the environment. The Workcell Controller (*WC*) receives a command from the environment to produce a number of products, then controls the operating of the other components and reports a ready-status back to the environment. So the total of the four components can be viewed as one workcell, producing and processing a number of products. The aim is to specify the components in such a way that the workcell behaves as desired.

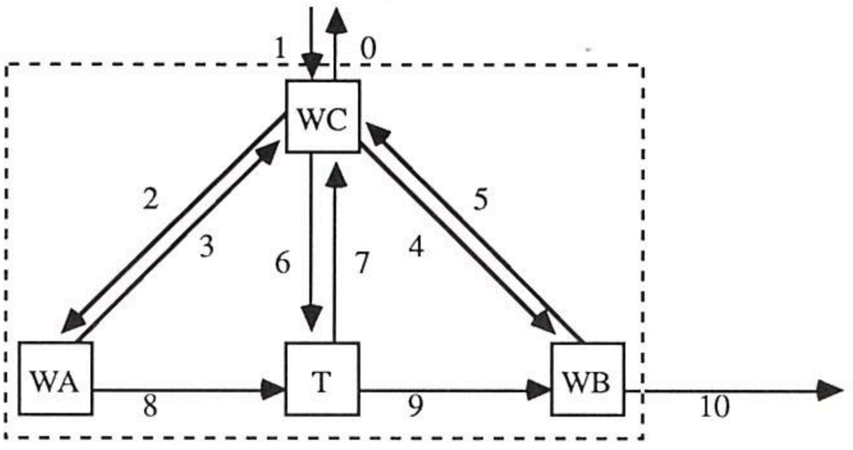

FIGURE 2. A simple workcell

2.1.2. Definitions. The four components are connected by 11 ports. Some ports are used to transmit data (the ports 0 through 7), while others are used to exchange products (the ports 8 through 10). Three ports are connected to the environment (the ports 0, 1 and 10). The set P of Ports is defined by

$$P = \{0, 1, ..., 10\}.$$

The set $PROD$ contains all products that are produced and processed within the workstation (or the complete factory). It contains the products product1 ($p\,1$) and the processed product1 ($proc(p\,1)$), but could contain other products as well.

$$PROD \supseteq \{p\,1,\ proc(p\,1)\}$$

Different kinds of data have to be transmitted. Via the ports 1, 2 and 4 a non-negative integer (n) can be sent to indicate that the receiver has to produce (or process) n products. We assume that this number has some upper bound N, which determines the maximum number of products the workcell can deal with in one drive. One can consider this number as a parameter of the specification. A ready message (r) is sent back over the ports 0, 3 and 5 to indicate that the component has fulfilled its task. Over port 6 the Workcell Controller can send a Transport Command (tc) to the Transport service, indicating that one product has to be transported from WA to WB. If this is done, an arrival-message (ar) is sent back via port 7. The ports 8 and 9 are used to transmit product1 ($p\,1$) and port 10 is used to transmit the processed product1 ($proc(p\,1)$). So the set of items that can be transmitted (D) is defined as

$$D = \{n \mid 0 \leqslant n \leqslant N\} \cup \{r, tc, ar\} \cup PROD.$$

A component can offer some element d of D at some port p by executing a send action ($sp(d)$). If simultaneously another component is able to execute a read action ($rp(d)$) at the same port and with the same element of D, this element is communicated ($cp(d)$) via port p. In this way both products and information will be distributed through the workcell. The atomic action t is used to denote an internal action, which will not be visible to the environment. The set of all atomic actions that can be performed is defined by

$$A = \{sp(d), rp(d), cp(d) \mid p \in P \wedge d \in D\} \cup \{t\}.$$

The communication function on the atoms is defined by

$$rp(d) | sp(d) = sp(d) | rp(d) = cp(d) \text{ for } p \in P \text{ and } d \in D.$$

All other communications yield deadlock.

Now we come to the definition of the four components.

2.1.3. Workstation A. Workstation A receives via port 2 the command to produce n times product $p\,1$ ($\Sigma_{n \geqslant 0} r\,2(n)$). Then it executes this command by producing n products (XA^n) and sends a ready-status message at port 3 ($s\,3(r)$). Then WA starts all over. If WA was commanded to produce zero products, XA^0 just ends after doing some internal action t. If a positive number of products has to be produced (XA^{n+1}), this is done by producing one product, followed by the production of n products (XA^n).

$$WA = \sum_{n \geq 0} (r2(n) \cdot XA^n \cdot s3(r) \cdot WA)$$

$$XA^0 = t$$

$$XA^{n+1} = s8(p1) \cdot XA^n$$

2.1.4. Workstation B. Workstation B has almost the same definition as Workstation A. It accepts the command to process n products via port 4 ($\Sigma_{n \geq 0} r4(n)$), processes n products (XB^n), sends a ready-status message ($s5(r)$) and starts all over. The processing of n products is achieved by repeatedly receiving an arbitrary product p at port 9 ($\Sigma_{p \in PROD} r9(p)$) and sending the processed version of this product to port 10 ($s10(proc(p))$).

$$WB = \sum_{n \geq 0} (r4(n) \cdot XB^n \cdot s5(r) \cdot WB)$$

$$XB^0 = t$$

$$XB^{n+1} = \sum_{p \in PROD} (r9(p) \cdot s10(proc(p)) \cdot XB^n)$$

2.1.5. Transport service. The Transport service (T) can be seen as a FIFO-queue. It is indexed with its contents. Adding an element p to the queue with contents σ, yields the queue with as its contents the concatenation $p*\sigma$. The empty queue is denoted by λ. The transport system either has an empty queue, or contains elements. If the queue is empty, T can receive a transport-command via port 6 ($r6(tc)$) and then it receives some product via port 8 ($\Sigma_{p \in PROD} r8(p)$). Next the transport service behaves as the transport service with one element in its queue (T^p). It is also possible to receive the product first and then receive the transport-command. If the queue was not empty, the Transport service has both options as mentioned for the empty queue, but it also has the option to send an element out of the queue at port 9 ($s9(q)$). Then the arrival of this element is reported to the Workcell Controller ($s7(ar)$) and the element is deleted from the queue.

$$T^\lambda = r6(tc) \cdot \sum_{p \in PROD} (r8(p) \cdot T^p) + \sum_{p \in PROD} (r8(p) \cdot r6(tc) \cdot T^p)$$

$$T^{\sigma * q} = r6(tc) \cdot \sum_{p \in PROD} (r8(p) \cdot T^{p * \sigma * q}) +$$

$$+ \sum_{p \in PROD} (r8(p) \cdot r6(tc) \cdot T^{p * \sigma * q}) + s9(q) \cdot s7(ar) \cdot T^\sigma$$

2.1.6. Workcell Controller. The Workcell Controller (WC) controls the communication with the environment and the interaction of the other components. It receives via port 1 the command to produce and process n products ($\Sigma_{n \geqslant 0} r1(n)$). Then it commands Workstation B to process n products ($s4(n)$) and goes into state D^n were n times product1 is produced and transported. Then finally it receives a ready-status message from WB via port 5 ($r5(r)$) and sends *ready* to the environment ($s0(r)$), returning to its initial state. The production and transport of n products is done in D^n. It repeatedly commands via port 2 Workstation A to produce one single product ($s2(1)$). If this is done a ready message is received at port 3 ($r3(r)$) and a transport command is sent at port 6 ($s6(tc)$). If the product has arrived at Workstation B, an arrival message is received at port 7 ($r7(ar)$).

$$WC = \sum_{n \geqslant 0} (r1(n) \cdot s4(n) \cdot D^n \cdot r5(r) \cdot s0(r) \cdot WC)$$

$$D^0 = t$$

$$D^{n+1} = s2(1) \cdot r3(r) \cdot s6(tc) \cdot r7(ar) \cdot D^n$$

2.1.7. The workcell. The concurrent operation of these four components can be considered as the specification of the whole workcell:

$$WC \| T^\lambda \| WA \| WB.$$

Notice that the Transport service has to start with an empty queue.

Of course all unsuccessful communications must be encapsulated, so define the encapsulation set H to contain all internal send and receive actions:

$$H = \{rp(d), sp(d) | 2 \leqslant p \leqslant 9 \wedge d \in D\}.$$

Furthermore we are only interested in the external behaviour of the system, so define

$$I = \{cp(d) | 2 \leqslant p \leqslant 9 \wedge d \in D\} \cup \{t\}.$$

Now the complete definition of the Workcell (W) is

$$\boxed{W = \tau_I \partial_H (WC \| T^\lambda \| WA \| WB)}$$

SPECIFICATION 2.1

2.2. Correctness

When designing the workcell, we had in mind some idea about its external behaviour. It receives a command at port 1, which indicates the number of products that has to be produced, then these products are produced and offered at port 10 and finally a ready message is offered at port 0 and we return to the starting state. This intended behaviour can easily be specified:

$$\boxed{\begin{aligned} V &= \sum_{n \geqslant 0}(r\,1(n)\cdot E^n\cdot V) \\ E^0 &= s\,0(r) \\ E^{n+1} &= s\,10(proc\,(p\,1))\cdot E^n \end{aligned}}$$

SPECIFICATION 2.2

Now, using RDP, let v and w be solutions of the two given specifications, 2.1 and 2.2. A proof that the processes v and w are equal can be seen as a verification that the specification of W is correct with respect to its intended external behaviour.

THEOREM 2.3. *The specification of the workcell is correct.*

$$ACP_\tau + RDP + RSP + ET \vdash v = w$$

PROOF. The proof consists of a series of successive expansions. All atoms that do not communicate yield deadlock, because they are encapsulated. The atoms that do communicate are underlined. All actions that are not abstracted from are boldfaced.

$$\begin{aligned} W &= \tau_I \partial_H (WC \| T^\lambda \| WA \| WB) \\ &= \tau_I \partial_H ((\sum \underline{r\,1(n)}\cdot s\,4(n)\cdot D^n \cdot r\,5(r)\cdot s\,0(r)\cdot WC)\| \\ &\qquad (r\,6(tc)\cdot \sum(r\,8(p)\cdot T^p) + \sum(r\,8(p)\cdot r\,6(tc)\cdot T^p))\| \\ &\qquad (\sum r\,2(n)\cdot XA^n \cdot s\,3(r)\cdot WA)\| \\ &\qquad (\sum r\,4(n)\cdot XB^n \cdot s\,5(r)\cdot WB)) \\ &= \sum \mathbf{r1(n)}\cdot \tau_I \partial_H ((\underline{s\,4(n)}\cdot D^n \cdot r\,5(r)\cdot s\,0(r)\cdot WC)\| \\ &\qquad (r\,6(tc)\cdot \sum(r\,8(p)\cdot T^p) + \sum(r\,8(p)\cdot r\,6(tc)\cdot T^p))\| \\ &\qquad (\sum r\,2(n)\cdot XA^n \cdot s\,3(r)\cdot WA)\| \\ &\qquad (\sum \underline{r\,4(n)}\cdot XB^n \cdot s\,5(r)\cdot WB)) \\ &= \sum \mathbf{r1(n)}\cdot \tau_I (c\,4(n)\cdot \partial_H ((D^n \cdot r\,5(r)\cdot s\,0(r)\cdot WC)\| \\ &\qquad (r\,6(tc)\cdot \sum(r\,8(p)\cdot T^p) + \sum(r\,8(p)\cdot r\,6(tc)\cdot T^p))\| \\ &\qquad (\sum r\,2(n)\cdot XA^n \cdot s\,3(r)\cdot WA)\| \\ &\qquad (XB^n \cdot s\,5(r)\cdot WB))) \end{aligned}$$

Now let

$$
\begin{aligned}
K^n \;=\; & \tau_I\partial_H((D^n\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot\sum(r8(p)\cdot T^p)+\sum(r8(p)\cdot r6(tc)\cdot T^p))\|\\
& (\sum r2(n)\cdot XA^n\cdot s3(r)\cdot WA)\|\\
& (XB^n\cdot s5(r)\cdot WB)),\ \text{then}
\end{aligned}
$$

$$
\begin{aligned}
K^0 = & \tau_I\partial_H((t\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot\sum(r8(p)\cdot T^p)+\sum(r8(p)\cdot r6(tc)\cdot T^p))\|\\
& (\sum r2(n)\cdot XA^n\cdot s3(r)\cdot WA)\|\\
& (t\cdot s5(r)\cdot WB))\\
= & \tau\cdot\tau_I\partial_H((s0(r)\cdot WC)\|\\
& (r6(tc)\cdot\sum(r8(p)\cdot T^p)+\sum(r8(p)\cdot r6(tc)\cdot T^p))\|\\
& (\sum r2(n)\cdot XA^n\cdot s3(r)\cdot WA)\|\\
& (WB))\\
= & \tau\cdot\mathbf{s0(r)}\cdot W
\end{aligned}
$$

$$
\begin{aligned}
K^{n+1} \;=\; & \tau_I\partial_H((s2(1)\cdot r3(r)\cdot s6(tc)\cdot r7(ar)\cdot D^n\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot\sum(r8(p)\cdot T^p)+\sum(r8(p)\cdot r6(tc)\cdot T^p))\|\\
& (\sum r2(n)\cdot XA^{n+1}\cdot s3(r)\cdot WA)\|\\
& (\sum r9(p)\cdot s10(proc(p))\cdot XB^n\cdot s5(r)\cdot WB))\\
= & \tau_I(c2(1)\cdot\partial_H((r3(r)\cdot s6(tc)\cdot r7(ar)\cdot D^n\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot\sum(r8(p)\cdot T^p)+\sum(r8(p)\cdot r6(tc)\cdot T^p))\|\\
& (s8(p1)\cdot XA^0\cdot s3(r)\cdot WA)\|\\
& (\sum r9(p)\cdot s10(proc(p))\cdot XB^n\cdot s5(r)\cdot WB)))\\
= & \tau\cdot\tau_I(c8(p1)\cdot\partial_H((r3(r)\cdot s6(tc)\cdot r7(ar)\cdot D^n\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot T^{p1})\|\\
& (t\cdot s3(r)\cdot WA)\|\\
& (\sum r9(p)\cdot s10(proc(p))\cdot XB^n\cdot s5(r)\cdot WB)))\\
= & \tau\cdot\tau_I(c3(r)\cdot\partial_H((s6(tc)\cdot r7(ar)\cdot D^n\cdot r5(r)\cdot s0(r)\cdot WC)\|\\
& (r6(tc)\cdot T^{p1})\|\\
& (WA)\|\\
& (\sum r9(p)\cdot s10(proc(p))\cdot XB^n\cdot s5(r)\cdot WB)))
\end{aligned}
$$

$$\begin{aligned}
(^{**}) &= \tau\cdot\tau_I(c\,6(tc)\cdot\partial_H((r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^{p^*p\,1})+\sum(r\,8(p)\cdot r\,6(tc)\cdot T^{p^*p\,1})+\underline{s\,9(p\,1)}\cdot s\,7(ar)\cdot T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (\underline{\textstyle\sum r\,9(p)}\cdot s\,10(proc\,(p))\cdot XB^n\cdot s\,5(r)\cdot WB)))\\
&= \tau\cdot\tau_I(c\,9(p\,1)\cdot\partial_H((\underline{r\,7(ar)}\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (\underline{s\,7(ar)}\cdot T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^n\cdot s\,5(r)\cdot WB)))\\
&= \tau((\tau_I(c\,7(ar)\cdot\partial_H((D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (s\,10(proc\,(p\,1))\cdot XB^n\cdot s\,5(r)\cdot WB)))+\\
&\qquad +\mathbf{s10(proc(p1))}\cdot K^n)
\end{aligned}$$

Now let

$$\begin{aligned}
L^n &= \tau_I\partial_H((D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (s\,10(proc\,(p\,1))\cdot XB^n\cdot s\,5(r)\cdot WB)),\ \text{then}\\
L^0 &= \tau_I\partial_H((t\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot t\cdot s\,5(r)\cdot WB))\\
&= \mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((t\cdot \underline{r\,5(r)}\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (\tau\cdot \underline{s\,5(r)}\cdot WB))+\\
&\quad \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((\underline{r\,5(r)}\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (t\cdot \underline{s\,5(r)}\cdot WB))
\end{aligned}$$

$$
\begin{aligned}
&= \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I(c\,5(r)\cdot\partial_H((\mathbf{s0(r)}\cdot WC)\| && \text{[using T2]}\\
&\qquad (T^{\lambda})\|\\
&\qquad (WA)\|\\
&\qquad (WB)))\\
&= \tau\cdot\mathbf{s10(proc(p1))}\cdot\mathbf{s0(r)}\cdot W\\
L^{n+1} &= \tau_I\partial_H((\underline{s\,2(1)}\cdot r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(r\,8(p)\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\textstyle\sum\underline{r\,2(n)}\cdot XA^n\cdot s\,3(r)\cdot WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^{n+1}\cdot s\,5(r)\cdot WB))\\
&= \tau_I(c\,2(1)\cdot\partial_H((r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(\underline{r\,8(p)}\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\underline{s\,8(p\,1)}\cdot XA^0\cdot s\,3(r)\cdot WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^{n+1}\cdot s\,5(r)\cdot WB)))+\\
&\quad \mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((\underline{s\,2(1)}\cdot r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(r\,8(p)\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\textstyle\sum\underline{r\,2(n)}\cdot XA^n\cdot s\,3(r)\cdot WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB))\\
&= \tau\cdot\tau_I(c\,8(p\,1)\cdot\partial_H((r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot T^{p\,1})\|\\
&\qquad (XA^0\cdot s\,3(r)\cdot WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^{n+1}\cdot s\,5(r)\cdot WB)))+\\
&\quad \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(\underline{r\,8(p)}\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\underline{s\,8(p\,1)}\cdot XA^0\cdot s\,3(r)\cdot WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB))+\\
&\quad \mathbf{s10(proc(p1))}\cdot\tau_I(c\,2(1)\cdot\partial_H((r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(\underline{r\,8(p)}\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\underline{s\,8(p\,1)}\cdot XA^0\cdot s\,3(r)\cdot WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB)))
\end{aligned}
$$

(The first two summands in this expression come from the first summand in the previous expression. Axiom T2 states that the summation of the second and third summand equals the second summand.)

$$\begin{aligned}
&= \tau\cdot\tau_I(c\,8(p\,1)\cdot\partial_H((\underline{r\,3(r)}\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot T^{p\,1})\|\\
&\qquad (t\cdot \underline{s\,3(r)}\cdot WA)\|\\
&\qquad (s\,10(proc\,(p\,1))\cdot XB^{n+1}\cdot s\,5(r)\cdot WB)))+\\
&\quad \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((r\,3(r)\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot\textstyle\sum(r\,8(p)\cdot T^p)+\sum(\underline{r\,8(p)}\cdot r\,6(tc)\cdot T^p))\|\\
&\qquad (\underline{s\,8(p\,1)}\cdot XA^0\cdot s\,3(r)\cdot WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB))\\
&= \tau\cdot\tau_I(c\,3(r)\cdot\partial_H((\underline{s\,6(tc)}\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (\underline{r\,6(tc)}\cdot T^{p\,1})\|\\
&\qquad (WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^{n+1}\cdot s\,5(r)\cdot WB)))+\\
&\quad \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I(c\,8(p\,1)\cdot\partial_H(\underline{r\,3(r)}\cdot s\,6(tc)\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (r\,6(tc)\cdot T^{p\,1})\|\\
&\qquad (t\cdot \underline{s\,3(r)}\cdot WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB)))\\
&= \tau\cdot\tau_I(c\,6(tc)\cdot\partial_H((r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{p\,1})\|\\
&\qquad (WA)\|\\
&\qquad (\underline{s\,10(proc\,(p\,1))}\cdot XB^{n+1}\cdot s\,5(r)\cdot WB)))+\\
&\quad \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I(c\,3(r)\cdot\partial_H((\underline{s\,6(tc)}\cdot r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (\underline{r\,6(tc)}\cdot T^{p\,1})\|\\
&\qquad (WA)\|\\
&\qquad (XB^{n+1}\cdot s\,5(r)\cdot WB)))\\
&= \tau\cdot\mathbf{s10(proc(p1))}\cdot\tau_I\partial_H((r\,7(ar)\cdot D^n\cdot r\,5(r)\cdot s\,0(r)\cdot WC)\|\\
&\qquad (T^{p\,1})\|\\
&\qquad (WA)\|\\
&\qquad (\textstyle\sum r\,9(p)\cdot s\,10(proc\,(p))\cdot XB^n\cdot s\,5(r)\cdot WB))\\
&= \tau\cdot\mathbf{s10(proc(p1))}\cdot K^{n+1} \qquad\qquad \text{[see(**)]}
\end{aligned}$$

So the process w is a solution of the following system:

$$\boxed{\begin{aligned}
&W=\sum r\,1(n)\cdot K^n\\
&K^0=\tau\cdot s\,0(r)\cdot W\\
&K^{n+1}=\tau(\tau\cdot L^n+s\,10(proc\,(p\,1))\cdot K^n)\\
&L^0=\tau\cdot s\,10(proc\,(p\,1))\cdot s\,0(r)\cdot W\\
&L^{n+1}=\tau\cdot s\,10(proc\,(p\,1))\cdot K^{n+1}
\end{aligned}}$$

SPECIFICATION 2.3

Now look at the specification of the process V, which specifies the intended behaviour. From RDP it follows that a solution (v, e^n) exists. Now, if v is also a solution of the specification for W, RSP can be used to infer that v equals w.

Define k^n and l^n by:

$$\begin{aligned}
k^n &= \tau\cdot e^n\cdot v\\
l^n &= \tau\cdot e^{n+1}\cdot v, \text{ then}\\
v &= \sum r\,1(n)\cdot e^n\cdot v=\sum r\,1(n)\cdot k^n\\
k^0 &= \tau\cdot e^0\cdot v=\tau\cdot s\,0(r)\cdot v\\
k^{n+1} &= \tau\cdot e^{n+1}\cdot v=\tau(\tau\cdot e^{n+1}\cdot v+e^{n+1}\cdot v)=\tau(\tau\cdot l^n+s\,10(proc\,(p\,1)\cdot e^n\cdot v)\\
&= \tau\cdot(\tau\cdot l^n+s\,10(proc\,(p\,1))\cdot k^n)\\
l^0 &= \tau\cdot e^1\cdot v=\tau\cdot s\,10(proc\,(p\,1))\cdot e^0\cdot v=\tau\cdot s\,10(proc\,(p\,1))\cdot s\,0(r)\cdot v\\
l^{n+1} &= \tau\cdot e^{n+2}\cdot v=\tau\cdot s\,10(proc\,(p\,1))\cdot e^{n+1}\cdot v=\tau\cdot s\,10(proc\,(p\,1))\cdot l^n
\end{aligned}$$

So (v,k^n,l^n) is a solution of specification 2.3.

2.4. Redundancy

Note that the specification of the workcell contains some redundancy. Although the transport service has the capability to store any number of products in the queue, this feature is not used in the workcell. At any moment not more than one product is stored in the buffer. So a one-item buffer would have functioned in the same way. Also, the option of receiving first a transport command and then a product is not used.

The capability of workstation A to receive a command to produce more than one product is also not used.

3. A Workcell with quality check

3.1. Specification

3.1.1. Global description. In this section a more complex workcell will be defined, having the possibility of checking the quality of the produced goods. Again we assume that some upper bound N is given which is the maximum number of products the workcell can produce in one drive. The workcell consists of four components. (1) Workstation A (WA) accepts a product, processes it and returns either a good product or a faulty product. (2) The Transport service (T) is a queue, at the one end accepting and at the other end sending products. After receiving a product, the (3) Quality check (Q) determines whether it is a good product or not. A good product will be passed along, while a rejected product will be removed. The latter occurrence is signaled to the (4) Workcell Controller (WC). This part controls the workcell. It receives the number of products that have to be processed, and instructs the workcell to do so. While the processing is going on, it will count the number of rejected products. At the end the workcell is instructed to process again an amount of products, equal to the number of rejections.

The workcell is graphically depicted in Figure 3.

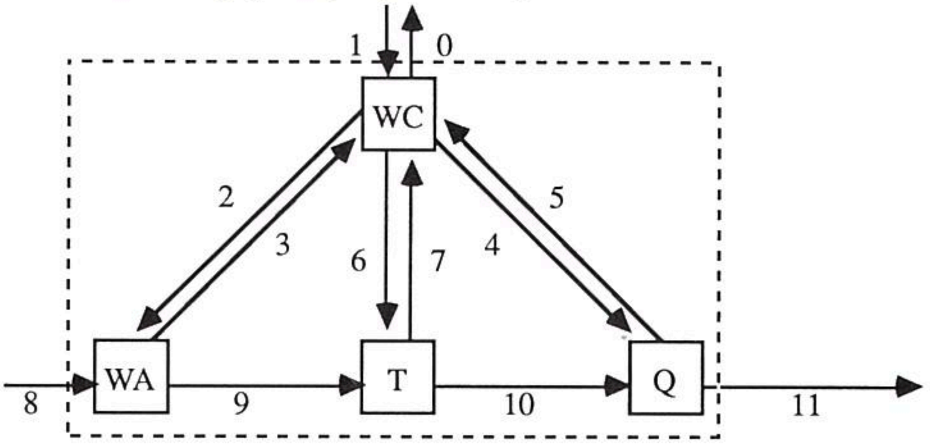

FIGURE 3. A workcell with quality check

3.1.2. Definitions. The four components are connected to each other by 12 ports. The ports 0 through 7 are used to transmit data and the ports 8 through 11 are used to exchange products. The ports 0, 1, 8 and 11 are connected to the environment. The set P of Ports is defined by

$$P = \{0, 1, ..., 11\}.$$

The set $PROD$ contains all products that are produced and processed within the workstation. It contains product1 ($p1$) and the product $p1$ after either good or faulty processing ($proc(p1, ok)$ and $proc(p1, fault)$).

$$PROD \supseteq \{p1,\ proc(p1,ok),\ proc(p1,fault)\}$$

A partial function *qual* can determine whether the processing of a product has been good or faulty.

$$qual(proc(p1,ok))=ok$$
$$qual(proc(p1,fault))=fault$$

Note that the information about the quality of a processed product is attached to the product itself, and one can only become aware of it by explicitly using the *qual* function. As an example consider drilling a hole in some product. After drilling, the hole is in the right position or not, but one can only become aware of this after applying some measuring tool, which reveals the quality. Along ports 1, 2, 4 and 6 a non-negative integer (n) can be sent to indicate that the receiver has to cope with n products. A ready message (r) is sent back over the ports 0, 3, 5 and 7 to indicate that the component has fulfilled its task. Port 5 is also used to indicate that a product has been rejected (rej). So the set D, of items that can be transmitted is defined as

$$D=\{n|0\leqslant n\leqslant N\}\cup\{r,rej\}\cup PROD.$$

Thus the set of atomic actions can be defined by:

$$A=\{sp(d),rp(d),cp(d)|p\in P\wedge d\in D\}\cup\{i\}.$$

The atom i is used to indicate an internal action. The communication function on atoms is defined by

$$rp(d)|sp(d)=sp(d)|rp(d)=cp(d)\quad \text{for } p\in P \text{ and } d\in D.$$

All other communications yield deadlock.

After these preliminary definitions we come to the specification of the four components.

3.1.3. Workstation A. Workstation A is a machine able to process a specified number of products. This number is received over port 2 ($\Sigma_{n\geqslant 0} r2(n)$). Then it executes its function n times (XA^n). The process XA^0 simply sends a ready message ($s3(r)$) and starts the workstation all over. The process XA^{n+1} is able to receive some product ($\Sigma_{p\in PROD} r8(p)$), which has to be processed. The possibility of either doing a good job or making an error while processing, is modeled by using the nondeterministic choice operator. By prefixing the actions with the internal atom i, a choice is made which cannot be influenced by the environment.

$$WA=\sum_{n\geqslant 0} r2(n)\cdot XA^n$$

$$XA^0=s3(r)\cdot WA$$

$$XA^{n+1}=\sum_{p\in PROD}(r8(p)\cdot(i\cdot s9(proc(p,ok))+i\cdot s9(proc(p,fault))))\cdot XA^n$$

3.1.4. Transport service. The transport service can best be seen as a bounded FIFO-queue. First it receives the number of products that have to be transported ($\Sigma_{n \geqslant 0} r\,6(n)$). Then it behaves like the empty queue with bound n (T_n^λ). After transporting n products (T_0^λ) a ready message is sent to the controller ($s\,7(r)$) and it starts all over. The process T_n^σ is intended to model a queue with contents σ, where n denotes the number of products that still have to be read in to the queue. T_{n+1}^λ has an empty buffer, so it can only read in products ($\Sigma_{p \in PROD} r\,9(p)$). $T_0^{\sigma * q}$ can only output the contents of its buffer. The process $T_{n+1}^{\sigma * q}$ can either accept some product ($\Sigma_{p \in PROD} r\,9(p)$) or it can send a queued item ($s\,10(q)$). This transport service differs from the one defined in the previous section in the sense that it needs less external control and that the capability of buffering more than one product is being used. Also, its specification has less redundancy.

$$T = \sum_{n \geqslant 0} r\,6(n) \cdot T_n^\lambda$$

$$T_0^\lambda = s\,7(r) \cdot T$$

$$T_{n+1}^\lambda = \sum_{p \in PROD} (r\,9(p) \cdot T_n^p)$$

$$T_0^{\sigma q * q} = s\,10(q) \cdot T_0^\sigma$$

$$T_{n+1}^{\sigma q * q} = \sum_{p \in PROD} (r\,9(p) \cdot T_n^{p * \sigma * q}) + s\,10(q) \cdot T_{n+1}^\sigma$$

3.1.5. Quality check. The quality of the processed product is tested by the process Q. It receives the command to test n products ($\Sigma_{n \geqslant 0} r\,4(n)$). Then the n tests are performed (XQ^n). If there are no tests left to do (XQ^0) a ready message is sent back ($s\,5(r)$) and the quality check returns to its initial state. The checks are done by accepting some product ($\Sigma_{p \in PROD} r\,10(p)$) and determining the quality of that product ($XQ^n_{p,qual(p)}$). If the quality is *ok* then the product can continue on its way ($s\,11(p)$). If the quality is *fault* then a rejection message is sent to the workcell controller ($s\,5(rej)$) and the product is rejected (i.e. discarded).

$$Q = \sum_{n \geqslant 0} r\,4(n) \cdot XQ^n$$

$$XQ^0 = s\,5(r) \cdot Q$$

$$XQ^{n+1} = \sum_{p \in PROD} r\,10(p) \cdot XQ^n_{p,qual(p)}$$

$$XQ^n_{p,ok} = s\,11(p) \cdot XQ^n$$

$$XQ^n_{p,fault} = s\,5(rej) \cdot XQ^n$$

3.1.6. Workcell Controller. The workcell is controlled by the Workcell Controller. It receives the message to process n products ($\Sigma_{n \geqslant 0} r\,1(n)$). When this is done ($D^0$), a ready message is reported ($s\,0(r)$) and the controller starts all over. The process D^{n+1} handles the processing of $n+1$ products. It sends the number of products that have to be processed to Workstation A ($s\,2(n+1)$), the Transport service ($s\,6(n+1)$) and the Quality check ($s\,4(n+1)$). Then it starts to count the number of rejections, starting with 0 (RC_0). The Rejection Counter will be incremented when it receives a rejection message ($r\,5(rej)$). When the Quality check, the Transport service and Workstation A respectively send their ready messages ($r\,5(r) \cdot r\,7(r) \cdot r\,3(r)$), the controller again commands the workcell to process some number of products (D^n). This new number of products is equal to the number of rejections encountered up to that moment.

$$WC = \sum_{n \geqslant 0} r\,1(n) \cdot D^n$$

$$D^0 = s\,0(r) \cdot WC$$

$$D^{n+1} = s\,4(n+1) \cdot s\,6(n+1) \cdot s\,2(n+1) \cdot RC_0$$

$$RC_n = r\,5(r) \cdot r\,7(r) \cdot r\,3(r) \cdot D^n + r\,5(rej) \cdot RC_{n+1}$$

Note that the order in which the ready messages are received is of importance. If e.g. the ready message of WA can be received first, it is still possible for Q to contain faulty products. But then, since WC is not able to receive any rejection messages from Q, a deadlock would occur.

3.1.7. The workcell. Now we are interested in the parallel operation of the four components as described above:

$$WC \| T \| WA \| Q.$$

To filter out all unsuccessful communications we use the encapsulation operator. All unsuccessful communications are gathered in the set H:

$$H = \{rp(d), sp(d) | (p \in \{2,3,4,5,6,7,9,10\} \wedge d \in D\}.$$

Because we are only interested in the external behaviour of the system, we abstract from the internal actions and communications, and define

$$I = \{cp(d) \| p \in \{2,3,4,5,6,7,9,10\} \wedge d \in D\} \cup \{i\}.$$

Thus the final definition of the workcell W becomes

$$\boxed{W = \tau_I \partial_H (WC \| T \| WA \| Q)}$$

SPECIFICATION 3.1

3.2. Correctness

Now we have to define some criterion for correctness of the specification. It is not enough to require that for any command n along port 1 the workcell processes n products correctly and reports a ready message. The problem is that if there is not enough supply of products along port 8, the workcell can reach a deadlock situation, waiting for more products. So we will only consider the behaviour of the workcell in an environment, supplying an unlimited number of products. The supplier is repeatedly sending product $p\,1$ along port 8, and is defined by

$$S \quad = s\,8(p\,1)\cdot S.$$

Of course we have to encapsulate unsuccessful communications over port 8 and abstract from successful communications over this port.

$$H' = \{rp\,(d), sp\,(d) | p = 8 \wedge d \in D\}$$
$$I' = \{cp\,(d) | p = 8 \wedge d \in D\}$$

So we will consider the behaviour of the following specification (See also Figure 4).

$$\boxed{W' = \tau_{I'}\partial_{H'}(S \| W)}$$

SPECIFICATION 3.2.1

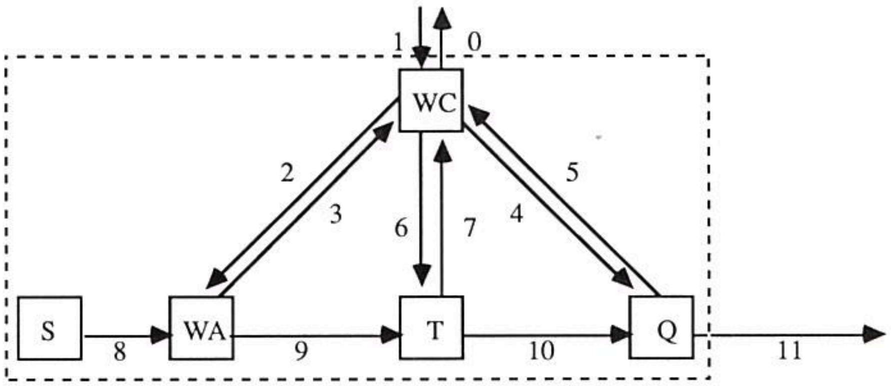

FIGURE 4. Adding a supplier to the workcell

The intended behaviour can be specified by the following specification 3.2.2. A command to process n products correctly will be received, then the n processed products will be delivered and a ready message will be reported.

$$V = \sum_{n \geq 0} (r\,1(n) \cdot E^n \cdot V)$$

$$E^0 = s\,0(r)$$

$$E^{n+1} = s\,11(proc(p\,1, ok)) \cdot E^n$$

SPECIFICATION 3.2.2

Now a verification of the correctness of the specification of the workcell will consist of a proof that specification 3.2.1. and specification 3.2.2. define the same process. So if w' and v are solutions of the two specifications, we have to prove $v = w'$.

THEOREM 3.3. *The specification of the workcell is correct.*

$$ACP_\tau + RDP + RSP + ET + CFAR + CA \vdash v = w'$$

PROOF

3.3.1. Step 1. First we reduce the number of components by aggregating the supplier S and workstation A. The resulting process (K) can be seen as being a supplier of either good or bad products (Figure 5).

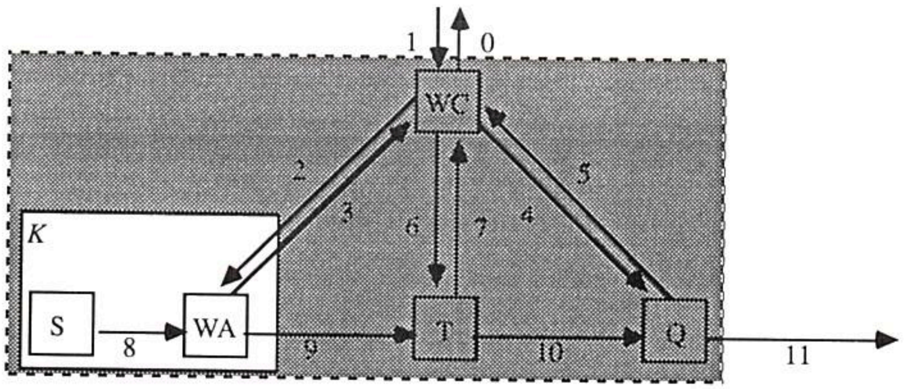

FIGURE 5. Aggregating S and WA

Let the process K be specified by

$$K = \sum r\,2(n) \cdot XK^n$$

$$XK^0 = s\,3(r) \cdot K$$

$$XK^{n+1} = (\tau \cdot s\,9(proc(p, ok)) + \tau \cdot s\,9(proc(p, fault))) \cdot XK^n.$$

And let the encapsulation set and the abstraction set be defined by

$$H1 = \{rp(d), sp(d) \mid p = 8 \wedge d \in D\},$$
$$I1 = \{cp(d) \mid p = 8 \wedge d \in D\} \cup \{i\},$$

then the following proposition holds:

PROPOSITION 3.3.1.1. $K = \tau_{I1}\partial_{H1}(S \| WA)$.

PROOF. Let the process L be defined by

$$\begin{aligned} L &= \tau_{I1}\partial_{H1}(S \| WA) \\ &= \tau_{I1}\partial_{H1}(S \| \textstyle\sum_{n \geq 0} r2(n) \cdot XA^n) \\ &= \textstyle\sum_{n \geq 0} r2(n) \cdot \tau_{I1}\partial_{H1}(S \| XA^n) \end{aligned}$$

Let L^n be defined by

$$\begin{aligned} L^n &= \tau_{I1}\partial_{H1}(S \| XA^n), \text{ then} \\ L^0 &= \tau_{I1}\partial_{H1}(S \| s3(r) \cdot WA) \\ &= s3(r) \cdot L \\ L^{n+1} &= \tau_{I1}\partial_{H1}(s8(p1) \cdot S \| \\ &\qquad \textstyle\sum_{p \in PROD}(r8(p) \cdot (i \cdot s9(proc(p,ok)) + i \cdot s9(proc(p,fault)))) \cdot XA^n) \\ &= \tau_{I1}(c8(p1) \cdot \partial_{H1}(S \| (i \cdot s9(proc(p1,ok)) + i \cdot s9(proc(p1,fault)))) \cdot XA^n) \\ &= \tau \cdot (\tau \cdot s9(proc(p1,ok)) + \tau \cdot s9(proc(p1,fault))) \cdot L^n \end{aligned}$$

Thus we have

$$L = \sum_{n \geq 0} r2(n) \cdot L^n$$
$$L^0 = s3(r) \cdot L$$
$$L^{n+1} = \tau \cdot (\tau \cdot s9(proc(p1,ok)) + \tau \cdot s9(proc(p1,fault))) \cdot L^n$$

Now it is easy to see that K and L define the same process. Use RSP to prove that a solution of K is also a solution of system L.

As a consequence of this proposition we can replace the two components S and WA by one simpler component K. This technique is called local replacement and was introduced in [12]. In order to actually replace the two components in the specification of the workcell, we need the conditional axioms (see [1]).

$$\begin{aligned} W' &= \tau_{I'}\partial_{H'}(S \| W) \\ &= \tau_{I'}\partial_{H'}(S \| \tau_I\partial_H(WA \| T \| Q \| WC)) \end{aligned}$$

$$=\tau_{I'\cup I}\partial_{H'\cup H}(S\|WA\|T\|Q\|WC)$$
$$=\tau_{I'\cup I}\partial_{H'\cup H}(\tau_{I1}\partial_{H1}(S\|WA)\|T\|Q\|WC)$$
$$=\tau_{I'\cup I}\partial_{H'\cup H}(K\|T\|Q\|WC)$$
$$=\tau_{I}\partial_{H}(K\|T\|Q\|WC)$$

3.3.2. Step 2. In the second step we will remove the parallelism in the specification by expanding the merges. This will result in a complex process, which describes all states that the workcell has.

First we define a new abstraction set, $I2$, obtained by deleting the communication of the rejection message from the old one. This will be useful when applying CFAR in step 3.

$$I2=I\setminus\{c5(rej)\}$$

If we define

$$U=\tau_{I2}\partial_H(K\|T\|Q\|WC), \text{then we have}$$
$$W'=\tau_{\{c5(rej)\}}(U).$$

For U we can derive

$$U=\tau_{I2}\partial_H(K\|T\|Q\|WC)=\sum_{n\geqslant 0} r1(n)\cdot\tau_{I2}\partial_H(K\|T\|Q\|D^n)$$

Let U^n be defined by $\tau_{I2}\partial_H(K\|T\|Q\|D^n)$, then

$$U^0=\tau_{I2}\partial_H(K\|T\|Q\|D^0)=s0(r)\cdot U$$
$$U^{n+1}=\tau_{I2}\partial_H(K\|T\|Q\|D^{n+1})$$
$$=\tau_{I2}(c4(n+1)\cdot c6(n+1)\cdot c2(n+1)\cdot\partial_H(XK^{n+1}\|T^{\lambda}_{n+1}\|XQ^{n+1}\|RC^0))$$

The process U^n denotes the total workcell, which has just received a command to produce a certain number of products. After distributing this command, the workcell enters the state in which the products will be produced. In the process of producing the products, there are several intermediate states. These states are determined by e.g. the number of products that still have to be produced, and the contents of the buffer of the transport service. The quality-check can also contain some product, i.e. the product which is read in and will be checked. All values that determine the actual state the workcell is in, are listed below:

choice	The choice made in K about processing correctly or faulty. The choice can be *ok* or *fault*. If no choice has been made yet, the value of this variable is $\times$.
count	The number of products that still have to be produced (not considering the number of rejected products).
buffer	The contents of the buffer in the transport service. The value is λ if the buffer is empty.

Qcont	The contents of the quality-check part. The value is λ if Q contains no product.
rc	The rejection counter, counting the number of rejected products.

All states can be described using these five variables. Now it is possible to define the process U, indexed by these five variables, which describes the behaviour of the workcell during the production of the products.

Define

$$U^{\text{choice, count, buffer, Qcont, rc}}$$

as the composition of the four components K, T, Q and WC, where the superscripts determine the state of the four components as follows:

If $choice = \times$ then K is in state XK^{count}, otherwise K is in state $s\,9(proc\,(p\,1, choice))\cdot XK^{\text{count}-1}$. T is in state $T^{\text{buffer}}_{\text{count}}$.

If $Qcont = \lambda$ then Q is in state $XQ^{\text{count}+|\text{buffer}|}$, otherwise Q is in state

$$XQ^{\text{count}+|\text{buffer}|}_{\text{Qcont,qual(Qcont)}}.$$

WC is in state RC_{count}.

For every combination of values we can calculate the behaviour of the system. Note that the choice can only be unequal to $\times$ if the count is positive. Let *ch* be some quality (i.e. either *ok* or *fault*), let n and rc be natural numbers, let σ be a series of processed products and let q be a processed product.

$U^{ch,n+1,\sigma,proc(p\,1,ok),rc}$

$$\begin{aligned}
&= \tau_I \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma}_{n+1} \,\|\, XQ^{n+1+|\sigma|}_{p,ok} \,\|\, RC_{rc})\\
&= \tau_I (c\,9(proc\,(p\,1,ch))\cdot \partial_H (XK^n \,\|\, T^{proc(p\,1,ch)^*\sigma}_n \,\|\, XQ^{n+1+|\sigma|}_{p,ok} \,\|\, RC_{rc}) +\\
&\quad s\,11(p)\cdot \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma}_{n+1} \,\|\, XQ^{n+1+|\sigma|} \,\|\, RC_{rc}))\\
&= \tau\cdot U^{\times,n,proc(p\,1,ch)^*\sigma,proc(p\,1,ok),rc} + s\,11(proc\,(p\,1,ok))\cdot U^{ch,n+1,\sigma,\lambda,rc}
\end{aligned}$$

$U^{ch,n+1,\sigma,proc(p\,1,fault),rc}$

$$\begin{aligned}
&= \tau_I \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma}_{n+1} \,\|\, XQ^{n+1+|\sigma|}_{p,fault} \,\|\, RC_{rc})\\
&= \tau_I (c\,9(proc\,(p\,1,ch))\cdot \partial_H (XK^n \,\|\, T^{proc(p\,1,ch)^*\sigma}_n \,\|\, XQ^{n+1+|\sigma|}_{p,fault} \,\|\, RC_{rc}) +\\
&\quad c\,5(rej)\cdot \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma}_{n+1} \,\|\, XQ^{n+1+|\sigma|} \,\|\, RC_{rc+1}))\\
&= \tau\cdot U^{\times,n,proc(p\,1,ch)^*\sigma,proc(p\,1,fault),rc} + c\,5(rej)\cdot U^{ch,n+1,\sigma,\lambda,rc+1}
\end{aligned}$$

$U^{ch,n+1,\sigma^*q,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma^*q}_{n+1} \,\|\, XQ^{n+2+|\sigma|} \,\|\, RC_{rc})\\
&= \tau_I (c\,9(proc\,(p\,1,ch))\cdot \partial_H (XK^n \,\|\, T^{proc(p\,1,ch)^*\sigma^*q}_n \,\|\, XQ^{n+2+|\sigma|} \,\|\, RC_{rc}) +\\
&\quad c\,10(q)\cdot \partial_H (s\,9(proc\,(p\,1,ch))\cdot XK^n \,\|\, T^{\sigma}_{n+1} \,\|\, XQ^{n+1+|\sigma|}_{q,qual(q)}\, RC_{rc}))\\
&= \tau\cdot U^{\times,n,proc(p\,1,ch)^*\sigma^*q,\lambda,rc} + \tau\cdot U^{ch,n+1,\sigma,q,rc}
\end{aligned}$$

$U^{ch,n+1,\lambda,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H(s\,9(proc(p\,1,ch))\cdot XK^n \| T^{\lambda}_{n+1} \| XQ^{n+1} \| RC_{rc})\\
&= \tau_I(c\,9(proc(p\,1,ch))\cdot\partial_H(XK^n \| T_n^{proc(p\,1,ch)} \| XQ^{n+1} \| RC_{rc}))\\
&= \tau\cdot U^{\times,n,proc(p\,1,ch),\lambda,rc}
\end{aligned}$$

$U^{\times,n+1,\sigma,proc(p\,1,ok),rc}$

$$\begin{aligned}
&= \tau_I \partial_H(XK^{n+1} \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,ok} \| RC_{rc})\\
&= \tau_I(\tau\cdot\partial_H(s\,9(proc(p\,1,ok))XK^n \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,ok} \| RC_{rc}) +\\
&\quad \tau\cdot\partial_H(s\,9(proc(p\,1,fault))XK^n \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,ok} \| RC_{rc}) +\\
&\quad s\,11(p)\cdot\partial_H(XK^{n+1} \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|} \| RC_{rc}))\\
&= \tau\cdot U^{ok,n+1,\sigma,proc(p\,1,ok),rc} + \tau\cdot U^{fault,n+1,\sigma,proc(p\,1,ok),rc} +\\
&\quad s\,11(proc(p\,1,ok))\cdot U^{\times,n+1,\sigma,\lambda,rc}
\end{aligned}$$

$U^{\times,n+1,\sigma,proc(p\,1,fault),rc}$

$$\begin{aligned}
&= \tau_I \partial_H(XK^{n+1} \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,fault} \| RC_{rc})\\
&= \tau_I(\tau\cdot\partial_H(s\,9(proc(p\,1,ok))XK^n \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,fault} \| RC_{rc}) +\\
&\quad \tau\cdot\partial_H(s\,9(proc(p\,1,fault))XK^n \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{p,fault} \| RC_{rc}) +\\
&\quad c\,5(rej)\cdot\partial_H(XK^{n+1} \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|} \| RC_{rc+1}))\\
&= \tau\cdot U^{ok,n+1,\sigma,proc(p\,1,fault),rc} + \tau\cdot U^{fault,n+1,\sigma,proc(p\,1,fault),rc} +\\
&\quad c\,5(rej)\cdot U^{\times,n+1,\sigma,\lambda,rc+1}
\end{aligned}$$

$U^{\times,n+1,\sigma*q,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H(XK^{n+1} \| T^{\sigma*q}_{n+1} \| XQ^{n+2+|\sigma|} \| RC_{rc})\\
&= \tau_I(\tau\cdot\partial_H(s\,9(proc(p\,1,ok))XK^n \| T^{\sigma*q}_{n+1} \| XQ^{n+2+|\sigma|} \| RC_{rc}) +\\
&\quad \tau\cdot\partial_H(s\,9(proc(p\,1,fault))XK^n \| T^{\sigma*q}_{n+1} \| XQ^{n+2+|\sigma|} \| RC_{rc}) +\\
&\quad c\,10(q)\cdot\partial_H(XK^{n+1} \| T^{\sigma}_{n+1} \| XQ^{n+1+|\sigma|}_{q,qual(q)} \| RC_{rc}))\\
&= \tau\cdot U^{ok,n+1,\sigma*q,\lambda,rc} + \tau\cdot U^{fault,n+1,\sigma*q,\lambda,rc} + \tau\cdot U^{\times,n+1,\sigma,q,rc}
\end{aligned}$$

$U^{\times,n+1,\lambda,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H(XK^{n+1} \| T^{\lambda}_{n+1} \| XQ^{n+1} \| RC_{rc})\\
&= \tau_I(\tau\cdot\partial_H(s\,9(proc(p\,1,ok))XK^n \| T^{\lambda}_{n+1} \| XQ^{n+1} \| RC_{rc}) +\\
&\quad \tau\cdot\partial_H(s\,9(proc(p\,1,fault))XK^n \| T^{\lambda}_{n+1} \| XQ^{n+1} \| RC_{rc}))\\
&= \tau\cdot U^{ok,n+1,\lambda,\lambda,rc} + \tau\cdot U^{fault,n+1,\lambda,\lambda,rc}
\end{aligned}$$

$U^{\times,0,\sigma,proc(p\,1,ok),rc}$

$$\begin{aligned}
&= \tau_I \partial_H (XK^0 \| T_0^\sigma \| XQ^{|\sigma|}_{p,ok} \| RC_{rc}) \\
&= \tau_I (s\,11(p) \cdot \partial_H (XK^0 \| T_0^\sigma \| XQ^{|\sigma|} \| RC_{rc})) \\
&= s\,11(proc(p\,1,ok)) \cdot U^{\times,0,\sigma,\lambda,rc}
\end{aligned}$$

$U^{\times,0,\sigma,proc(p\,1,fault),rc}$

$$\begin{aligned}
&= \tau_I \partial_H (XK^0 \| T_0^\sigma \| XQ^{|\sigma|}_{p,fault} \| RC_{rc}) \\
&= \tau_I (c\,5(rej) \cdot \partial_H (XK^0 \| T_0^\sigma \| XQ^{|\sigma|} \| RC_{rc+1})) \\
&= c\,5(rej) \cdot U^{\times,0,\sigma,\lambda,rc+1}
\end{aligned}$$

$U^{\times,0,\sigma * q,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H (XK^0 \| T_0^{\sigma * q} \| XQ^{|\sigma|+1} \| RC_{rc}) \\
&= \tau_I (c\,10(q) \cdot \partial_H (XK^0 \| T_0^\sigma \| XQ^{|\sigma|}_{q,qual(q)} \| RC_{rc})) \\
&= \tau \cdot U^{\times,0,\sigma,q,rc}
\end{aligned}$$

$U^{\times,0,\lambda,\lambda,rc}$

$$\begin{aligned}
&= \tau_I \partial_H (XK^0 \| T_0^\lambda \| XQ^0 \| RC_{rc}) \\
&= \tau_I (c\,5(r) \cdot c\,7(r) \cdot c\,3(r) \cdot \partial_H (K \| T \| Q \| D^{rc})) \\
&= \tau \cdot U^{rc}
\end{aligned}$$

Thus we have the following system:

1) $U = \sum_{n \geqslant 0} r\,1(n) \cdot U^n$

2) $U^0 = s\,0(r) \cdot U$

3) $U^{n+1} = \tau \cdot U^{\times, n+1, \lambda, \lambda, 0}$

4) $U^{ch, n+1, \sigma, proc(p\,1, ok), rc} =$
$= \tau \cdot U^{\times, n, proc(p\,1, ch) * \sigma, proc(p\,1, ok), rc} + s\,11(proc(p\,1, ok)) \cdot U^{uch, n+1, \sigma, \lambda, rc}$

5) $U^{ch, n+1, \sigma, proc(p\,1, fault), rc} =$
$= \tau \cdot U^{\times, n, proc(p\,1, ch) * \sigma, proc(p\,1, fault), rc} + c\,5(rej) \cdot U^{ch, n+1, \sigma, \lambda, rc+1}$

6) $U^{ch, n+1, \sigma * q, \lambda, rc} = \tau \cdot U^{\times, n, proc(p\,1, ch) * \sigma * q, \lambda, rc} + \tau \cdot U^{ch, n+1, \sigma, q, rc}$

7) $U^{ch, n+1, \lambda, \lambda, rc} = \tau \cdot U^{\times, n, proc(p\,1, ch), \lambda, rc}$

8) $U^{\times, n+1, \sigma, proc(p\,1, ok), rc} =$
$= \tau \cdot U^{ok, n+1, \sigma, proc(p\,1, ok), rc} + \tau \cdot U^{fault, n+1, \sigma, proc(p\,1, ok), rc} +$
$s\,11(proc(p\,1, ok)) \cdot U^{\times, n+1, \sigma, \lambda, rc}$

9) $U^{\times, n+1, \sigma, proc(p\,1, fault), rc} =$
$= \tau \cdot U^{ok, n+1, \sigma, proc(p\,1, fault), rc} + \tau \cdot U^{fault, n+1, \sigma, proc(p\,1, fault), rc} +$
$c\,5(rej) \cdot U^{\times, n+1, \sigma, \lambda, rc+1}$

10) $U^{\times, n+1, \sigma * q, \lambda, rc} =$
$= \tau \cdot U^{ok, n+1, \sigma * q, \lambda, rc} + \tau \cdot U^{fault, n+1, \sigma * q, \lambda, rc} + \tau \cdot U^{\times, n+1, \sigma, q, rc}$

11) $U^{\times, n+1, \lambda, \lambda, rc} = \tau \cdot U^{ok, n+1, \lambda, \lambda, rc} + \tau \cdot U^{fault, n+1, \lambda, \lambda, rc}$

12) $U^{\times, 0, \sigma, proc(p\,1, ok), rc} = s\,11(proc(p\,1, ok)) \cdot U^{\times, 0, \sigma, \lambda, rc}$

13) $U^{\times, 0, \sigma, proc(p\,1, fault), rc} = c\,5(rej) \cdot U^{\times, 0, \sigma, \lambda, rc+1}$

14) $U^{\times, 0, \sigma * q, \lambda, rc} = \tau \cdot U^{\times, 0, \sigma, q, rc}$

15) $U^{\times, 0, \lambda, \lambda, rc} = \tau \cdot U^{rc}$

SPECIFICATION 3.3.2

3.3.3. Step 3. In the final part of the proof we use CFAR (see [12]) and RSP (see [2]) to prove that the system derived in step 2 (specification 3.3.2) can be reduced to the desired specification V (specification 3.2.2).

Some observations about the specification above can be made. The number of products that still have to be produced correctly (m) can be determined from the values of the superscripts of the process:

$$\text{count} + |\text{buffer}| + |\text{Qcont}| + rc.$$

So we must prove the equality

$$\tau \cdot E^m = \tau \cdot \tau_{\{c\,5(rej)\}}(U^{\text{choice}, \text{count}, \text{buffer}, \text{Qcont}, \text{rc}})$$

for $m = count + |\text{buffer}| + |\text{Qcont}| + rc$. We must also prove

$$\tau \cdot E^m = \tau \cdot \tau_{\{c5(rej)\}}(U^m).$$

Comparing the two processes one easily notes that U^m has the possibility to produce only faulty products, hence it can loop forever, sending rejection messages. The process E^m however does not have this possibility. Thus we must make the assumption that workstation *WA* is not completely broken. It now and then must process some product correctly. This fairness assumption can be modeled in process algebra with the Cluster Fair Abstraction Rule.

The only cases in which it is possible to never process a product correctly are the processes which are indexed such that (i) choice$\neq ok$, (ii) the buffer contains no correctly processed products and (iii) Qcont$\neq proc(p1, ok)$. This observation leads us to consider clusters of processes which satisfy these conditions and have to produce the same number of products. Thus cluster m (for $m>0$) is defined by:

$$\begin{aligned} CL(m) = \{U^m\} \cup \{ & U^{\text{choice},\text{count},\text{buffer},\text{Qcont},rc} \mid \\ & choice \neq ok \wedge proc(p1, ok) \notin buffer \wedge \text{Qcont} \neq proc(p1, ok) \wedge \\ & count + |buffer| + |Qcont| + rc = m\}. \end{aligned}$$

This defines a conservative cluster from $\{c5(rej)\}$ in specification 3.3.2 (using terminology of [12]). The workcell can choose to loop forever in such a cluster, or it can choose to process some product correctly. This will be indicated by setting the *choice*-index to *ok*. After some time, this choice leads to a correctly processed product leaving the workcell. In the meantime the workcell has to make new choices. If they are all negative, we again enter a cluster that permits infinite loops. If a choice was made to produce one or more correct products, we are still in a state in which progress can be made.

Now we can determine the exits of such a cluster. These are all states which can be reached from the cluster, but are no member of it. Thus there are no correctly processed products in the buffers and the choice has been made to process the next product correctly.

$$\begin{aligned} EXITS(m) = & \{U^{ok,n+1,\sigma,proc(p1,fault),rc} \mid n+1+|\sigma|+1+rc = m \wedge proc(p1, ok) \notin \sigma\} \cup \\ & \{U^{ok,n+1,\sigma^* q,\lambda,rc} \mid n+1+|\sigma|+1+rc = m \wedge proc(p1, ok) \notin \sigma^* q\} \cup \\ & \{U^{ok,n+1,\lambda,\lambda,rc} \mid n+1+rc = m\} \end{aligned}$$

Applying CFAR to the specification derived in step 2 leads to a new specification. This specification is equal to the old one for states which contain some correctly processed products and is modified for states which only contain faulty products.

Now set

$$\begin{aligned} & W' = \tau_{\{c5(rej)\}}(U) \\ & W^n = \tau_{\{c5(rej)\}}(U^n) \\ & W^{\text{choice},\text{count},\text{buffer},\text{Qcont},rc} = \tau_{\{c5(rej)\}}(U^{\text{choice},\text{count},\text{buffer},\text{Qcont},rc}) \end{aligned}$$

In the first part of the following specification we assume that there are correctly processed products in the buffer σ, or in *Qcont*, or $ch = ok$. The numbers correspond to the numbers in the specification of *U*.

1) $W' = \Sigma_{n \geqslant 0} r1(n) \cdot W^n$

2) $W^0 = s0(r) \cdot W'$

4) $W^{ch,n+1,\sigma,proc(p1,ok),rc} =$
$= \tau \cdot W^{\times,n,proc(p1,ch)^{*}\sigma,proc(p1,ok),rc} + s11(proc(p1,ok)) \cdot W^{ch,n+1,\sigma,\lambda,rc}$

5) $W^{ch,n+1,\sigma,proc(p1,fault),rc} =$
$= \tau \cdot W^{\times,n,proc(p1,ch)^{*}\sigma,proc(p1,fault),rc} + \tau \cdot W^{ch,n+1,\sigma,\lambda,rc+1}$

6) $W^{ch,n+1,\sigma^{*}q,\lambda,rc} = \tau \cdot W^{\times,n,proc(p1,ch)^{*}\sigma^{*}q,\lambda,rc} + \tau \cdot W^{ch,n+1,\sigma,q,rc}$

7) $W^{ch,n+1,\lambda,\lambda,rc} = \tau \cdot W^{\times,n,proc(p1,ch),\lambda,rc}$

8) $W^{\times,n+1,\sigma,proc(p1,ok),rc} =$
$= \tau \cdot W^{ok,n+1,\sigma,proc(p1,ok),rc} + \tau \cdot W^{fault,n+1,\sigma,proc(p1,ok),rc} +$
$s11(proc(p1,ok)) \cdot W^{\times,n+1,\sigma,\lambda,rc}$

9) $W^{\times,n+1,\sigma,proc(p1,fault),rc} =$
$= \tau \cdot W^{ok,n+1,\sigma,proc(p1,fault),rc} + \tau \cdot W^{fault,n+1,\sigma,proc(p1,fault),rc} +$
$\tau \cdot W^{\times,n+1,\sigma,\lambda,rc+1}$

10) $W^{\times,n+1,\sigma^{*}q,\lambda,rc} =$
$= \tau \cdot W^{ok,n+1,\sigma^{*}q,\lambda,rc} + \tau \cdot W^{fault,n+1,\sigma^{*}q,\lambda,rc} + \tau \cdot W^{\times,n+1,\sigma,q,rc}$

12) $W^{\times,0,\sigma,proc(p1,ok),rc} = s11(proc(p1,ok)) \cdot W \times,0,\sigma,\lambda,rc$

13) $W^{\times,0,\sigma,proc(p1,fault),rc} = \tau \cdot W^{\times,0,\sigma,\lambda,rc+1}$

14) $W^{\times,0,\sigma^{*}q,\lambda,rc} = \tau \cdot W^{\times,0,\sigma,q,rc}$

SPECIFICATION 3.4.3, PART 1

In the second part we assume that there are no correct products in the workcell, so we are in a cluster. The expression $\sum EXITS(m)$ is shorthand for $\sum_{p \in EXITS(m)} \tau_{\{c5(rej)\}}(p)$.

$$\boxed{\begin{array}{ll}
3) & W^{n+1} = \tau\cdot\sum EXITS(n+1) \\
5a) & W^{ch,n+1,\sigma,proc(p1,fault),rc} = \tau\cdot\sum EXITS(n+1+|\sigma|+1+rc) \\
6a) & W^{ch,n+1,\sigma*q,\lambda,rc} = \tau\cdot\sum EXITS(n+1+|\sigma|+1+rc) \\
7a) & W^{ch,n+1,\lambda,\lambda,rc} = \tau\cdot\sum EXITS(n+1+rc) \\
9a) & W^{\times,n+1,\sigma,proc(p1,fault),rc} = \tau\cdot\sum EXITS(n+1+|\sigma|+1+rc) \\
10a) & W^{\times,n+1,\sigma*q,\lambda,rc} = \tau\cdot\sum EXITS(n+1+|\sigma|+1+rc) \\
11) & W^{\times,n+1,\lambda,\lambda,rc} = \tau\cdot\sum EXITS(n+1+rc) \\
13a) & W^{\times,0,\sigma,proc(p1,fault),rc} = \tau\cdot\sum EXITS(|\sigma|+1+rc) \\
14a) & W^{\times,0,\sigma*q,\lambda,rc} = \tau\cdot\sum EXITS(|\sigma|+1+rc) \\
15) & W^{\times,0,\lambda,\lambda,rc} = \tau\cdot\sum EXITS(rc)
\end{array}}$$

SPECIFICATION 3.4.3, PART 2

This specification now describes exactly the same process as specification 3.2.2. This can be easily verified by substituting V for W', E^0, for W^0, $\tau\cdot E^{n+1}$ for W^{n+1} and $\tau\cdot E^{count+|buffer|+|Qcont|+rc}$ for $W^{ch,count,buffer,Qcont,rc}$. Note that the only equation not starting with a τ is equation 12. So we must substitute $E^{|\sigma|+1}$ for $W^{\times,0,\sigma,proc(p1,ok)}$. So we see that V is a solution of the system defining W', and thus we can use RSP to conclude that V equals W'.

Note that RSP is only applicable if the specifications are guarded. A proof of the guardedness of specification 3.4.3 is straightforward.

4. FINAL REMARKS

The techniques introduced in this paper seem to be powerful enough to aid in the specification and verification of CIM-architectures. Although two workcells were considered of low complexity, the basic concepts of the technique are well illustrated. Now, due to the compositionality of the specifications, one can build a large plant consisting of a number of workcells which are already proved to function correctly. Thus, increasing the scale of the system will be possible.

It is also possible to add new features to the workcell and model them in process algebra. Possible features are: interrupts (modeled by the priority-operator, see [3]), detailed reports on the functioning of a machine, changing the tools of a machine, etc. Most of these features are not more complex than adding quality checks to a workcell.

Since a wide range of proof-rules and proof-techniques are developed in process algebra, the specification of a CIM-architecture in process algebra has advantages over specification in e.g. LOTOS. To name one, in LOTOS there is no equivalent of the fairness assumption.

ACKNOWLEDGEMENTS

I would like to express my thanks to Frank Biemans and Pieter Blonk of the Philips CAM centre for the fruitful discussions on this subject. I would also like to thank Jos Baeten, Jan Bergstra, Frits Vaandrager and Freek Wiedijk for proof reading and commenting on this paper.

REFERENCES

1. J.C.M. BAETEN, J.A. BERGSTRA, J.W. KLOP (1987). Conditional axioms and α/β calculus in process algebra. M. WIRSING (ed.). *Proc. IFIP Conf. on Formal Description of Programming Concepts - III,* Ebberup 1986, North-Holland, Amsterdam, 53-75.
2. J.C.M. BAETEN, J.A. BERGSTRA, J.W. KLOP (1987). On the consistency of Koomen's Fair Abstraction Rule. *Theoretical Computer Science 51 (1/2),* 129-176.
3. J.C.M. BAETEN, J.A. BERGSTRA, J.W. KLOP (1986). Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae IX(2),* 127-168.
4. J.A. BERGSTRA, J.W. KLOP (1986). Algebra of communicating processes. J.W. DE BAKKER, M. HAZEWINKEL, J.K. LENSTRA (eds.). *Mathematics and Computer Science,* CWI Monograph 1, North-Holland, Amsterdam, 89-138.
5. J.A. BERGSTRA, J.W. KLOP (1985). Algebra of communicating processes with abstraction. *Theoretical Computer Science 37(1),* 77-121.
6. J.A. BERGSTRA, J.W. KLOP (1984). Process algebra for synchronous communication. *Information and Control 60(1/3),* 109-137.
7. F. BIEMANS (1986). Reference model of production control systems. *Proc. of the IECON 86,* Milwaukee.
8. F. BIEMANS, P. BLONK (1986). On the formal specification and verification of CIM architectures using LOTOS. *Computers in Industry 7(6),* 491-504.
9. E. BRINKSMA (ed.) (1987). *LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour,* Report ISO DIS 8807.
10. H. KODATE, K. FUJII, K. YAMANOI (1987). Representation of FMS with petrinet graph and its application to simulation of system operation. *Robotics and Computer-Integrated Manufacturing 3(3),* 275-283.
11. N. KOMODA, K. KERA, T. KUBO (1984). An autonomous, decentralized control system for factory automation. *IEEE Trans. Comput 17(12),* 73-83.
12. F.W. VAANDRAGER (1984). *Verification of Two Communication Protocols by means of Process Algebra,* CWI Report CS-R8608, Centre for Mathematics and Computer Science, Amsterdam.