Wulfdieter L. Bauerfeld
Hahn-Meitner-Institut Berlin, Germany
on leave at
The University of Texas at Austin
Department of Computer Sciences

A Communication Concept for Protocol Models

A concept is introduced for the communication within hierarchically ordered distributed systems like computer network protocols. Characterstrings common for output and input instructions are sufficient to describe a broadcast communication via an unbuffered send. The broadcast can be restricted to clusters of processes to express the physical links in a network. Any hierarchical order is not described in process types but in messages which can be encoded or decoded.

1. Process-to-Process Communication

Protocols are considered as to be made up by at least two parallel processes whose execution agrees in certain rules and common variables in given formats. Modeling of protocols for verification, performance analysis or an adequate specification of the implementation demand a great deal from a communication method. The variety of methods is restricted to exchange of messages. Data sharing and parallelism are inherently difficult to combine effectively in a language [Qu79]. Shared storage locations can be modeled as a separate process. Thus communication solely by messages seems to be sufficient. Still different naming strategies used in message oriented communication systems have to be examined.

To send a message the receiving process can be named in the SEND instruction which is paired by a RECEIVE instruction where a sender description might be optional. This process naming requires that processes have access to each other names. An agreement upon the significance of the used variables should have been settled. However the number of messages sent or received can be different for both processes. Compiler type checking of messages to express different transition types is not possible. Individual SEND and RECEIVE instructions could be paired only if analysis of the behaviours of both processes is possible.

A more modular approach is to declare a number of ports for each module [Hu78]. Verification could be done by a technique known as 'symbolic execution' which covers all possible interactions between processes following a proof tree [BJ78]. It is assumed that individual processes define particular variables - the port names - to be accessable externally. Providing an external mechanism for linking which allows dynamic port linkage, any process name can be omitted. Ports can be of a given type; variables passing through a typed port must have a type that agrees with the port type. Checking of agreement of types between a message and a port name and between port names can be done before process execution begins [Qu79].