using some memory of its own, to produce output on
some or all of its output lines. It is assumed that:
i) Communication lines are the only way by which
computing stations may communicate.

ii) A communication line transmits information within
an unpredictable but finite amount of time.

Restrictions are imposed on the behaviour of com-
puting stations:

iii) At any given time, a computing station is either
computing or waiting for information on one of
its input lines.

iv) Each computing station follows a sequential
program. (We call here sequential program what is
usually called a program elsewhere).

Remarks: first, since several computing stations
may be computing simultaneously, this model indeed
exhibits some form of parallelism. Second, restric-
tion (iii) means that a computing station cannot be

Examples:
The process f of program G is associated to the
continuous function f in N^m x N^m -> N^m defined
recursively by:
f(0,0) = f(1,0)

The process g is associated to two functions, one per
output line, defined recursively by:
g_1(0) = (f(0,0), g_2(0,0)) and

g_2(0) = (f_2(0,0), g_3(0,0))

Similarly, the function h maps N into N :
h(x) = f(x,0) where x is in N and the notation
(s) means the unit length sequence whose first
element is s.

In these examples, not much computation is actually
performed on the inputs, but an arbitrary group of
(N.B.: this is a way to ensure that the parallel
expressive programs are syntactically well formed.)

6. DISCUSSION AND CONCLUSION
The kind of parallel programming we have studied in

REFERENCES.

[34] R. Atkinson and C. Hewitt. Synchronization in Actor
Systems. 4th SIGPLAN-SIGACT Symp. on Princ. of Prog.
Process SIFT (x QT = 00)

Pure PRIME

repeat

GET(00) = PRIME; PUT(PRIME,00)

GOOD FILTER(PRIME,00); CONTINUE close

endprocess

In our implementation, significant time and space savings result from this transform.

communication channels. Processes are represented by data structures containing local access environments and control continuations. A channel is represented by a linear list containing items stored in the channel and terminating with a reference to the current producer for the channel. Consumers have pointers into this list which are updated by the GET operation, while the producer inserts new items at the end of the list via the PUT operation. A reconfiguration instruction results in:

no need to deactivate the producer process; it may continue to run in anticipation of further demands for its output. In this way, computations that were interleaved in time can be made to overlap, and some process switching overhead is saved as well, without increasing the programmer's burden. The only drawback is that the process which was not deactivated may carry out nonessential computation, i.e., computation that is not needed to produce the final outcome of the program. This nonessential computation may even

Lemma 3: For any sequence \( L \), \( S_I^\Pi(L) \) is a subsequence of \( L \).

Lemma 4: If \( L \) is an increasing sequence and \( p \) occurs in \( S_I^\Pi(L) \), no other multiple of \( p \) occurs in \( S_I^\Pi(L) \).

Lemma 5: If every element of \( L \) is greater than 1 and if \( p \) is a prime occurring in \( L \), then \( p \) occurs in \( S_I^\Pi(L) \).

By lemmas 1 and 5, the output of the program must contain all primes. By lemma 4, composite numbers...

