type direction = enum(OUT, IN);
type template =
    rec(direction d; int source; int dest; int port);
type Templates = set of template;

chan match(Templates t);
chan reply[1:n](direction d; int who);
chan data[1:n](byte msg[*]);

output statement not in a guard:
    Templates t = template(OUT, myid, destination, port);
    send match(t);
    receive reply[myid](direction, who);
    # direction will be OUT and who will be destination
    gather expressions into a message buffer;
    send data[who](buffer);

input statement not in a guard:
    Templates t = template(IN, source, myid, port);
    send match(t);
    receive reply[myid](direction, who);
    # direction will be IN and who will be myid
    receive data[myid](buffer);
    unpack the buffer into local variables;

guarded input or output statement:
    Templates t = ∅;  # set of possible communications
    for [ boolean expressions in guards that are true ]
        insert a template for the input or output statement into set t;
    send match(t);      # send matches to clearing house
    receive reply[myid](direction, who);
    use direction and who to determine which guarded
    communication statement was the one that matched;
    if (direction == IN)
        { receive data[myid](buffer);
          unpack the buffer into local variables; }
    else  # direction == OUT
        { gather expressions into a message buffer;
          send data[who](buffer);  }
    execute appropriate guarded statement S;

Figure 10.7  Protocols for regular processes.