Algorithm 4: A part of ExternalRequest Algorithm

```
Input: CSP# atomic proc
if State(proc) is not Waiting then
    return false;
if proc is (Time-bound) SMT (and timestamp < proc.ddl) then // proc is (T)SMST or (T)SMRT
    if proc is SMST then
        emit message event of rcvProc; // rcvProc = Enable[proc[count]];
        set RcvEnabled[rcvProc] as true; // the receive condition is triggered
        stateChange(proc, Executing); // execute the task
        proc.count ++; // count refers to the number of the task instances
    if proc is SMRT and RcvEnabled[proc] == true then // execute the Receive Task
        stateChange(proc, Executing);
        proc.count ++;
if count == 1 and Inactivate[proc] exists then // at first call of this function
    for xorProc in Inactivate[proc] do // inactivate the xor procs
        stateChange(xorProc, Disabled);
if (count == InsNum) or (proc is time-bounded and count == proc.MsgNum) then // enough instances
    stateChange(proc, Done); // No more instances for this task if Activate[proc] exists then // Activate the next procs
        if Parallel[proc] exists and one of them is not Done then
            do nothing;
        else // next procs are permitted to be executed
            for nextProc in Activate[proc] do
                stateChange(nextProc, Waiting);
                if nextProc is time-bounded then
                    nextProc.ddl = (timestamp + nextProc.Duration);
else // instances not enough
    stateChange(proc, Waiting); // More instances of this task are permitted
```