VERIFICATION OF MPI PROGRAMS USING SESSION TYPES

K. Honda[1], E.R.B. Marques[2], F. Martins[2], N. Ng[3], V.T. Vasconcelos[2], N. Yoshida[3]

[1] Queen Mary, University of London [2] LaSIGE/FCUL, Universidade de Lisboa [3] Imperial College London

PROPOSAL

The **multiparty session type** methodology [3] considers the specification of a **global protocol** expressing interaction among multiple participants, from which an **endpoint protocol** can be derived (projected) for each individual participant, e.g., as in Scribble [2].

A **well-formed** (global) protocol can be verified in **polynomial time** and ensures by construction some key properties: **type safety**, **communication safety**, and **deadlock freedom** [3].

Our **aim** is to ensure these (paramount) properties for sound MPI programs by verifying (at compile-time) a **conformance relation** between an MPI program and a session type specification.

The **potential** is to overcome typical shortcomings of other state-of-the-art methodologies considered for MPI, e.g., model checking or symbolic execution [1, 4], that require program-level analysis for all properties of interest, and inherently lead to a **state-explosion problem** as the number of participants grows.

SAMPLE MPI PROGRAM

A simple (and naive) program loop with a pipeline communication pattern and a global reduction.

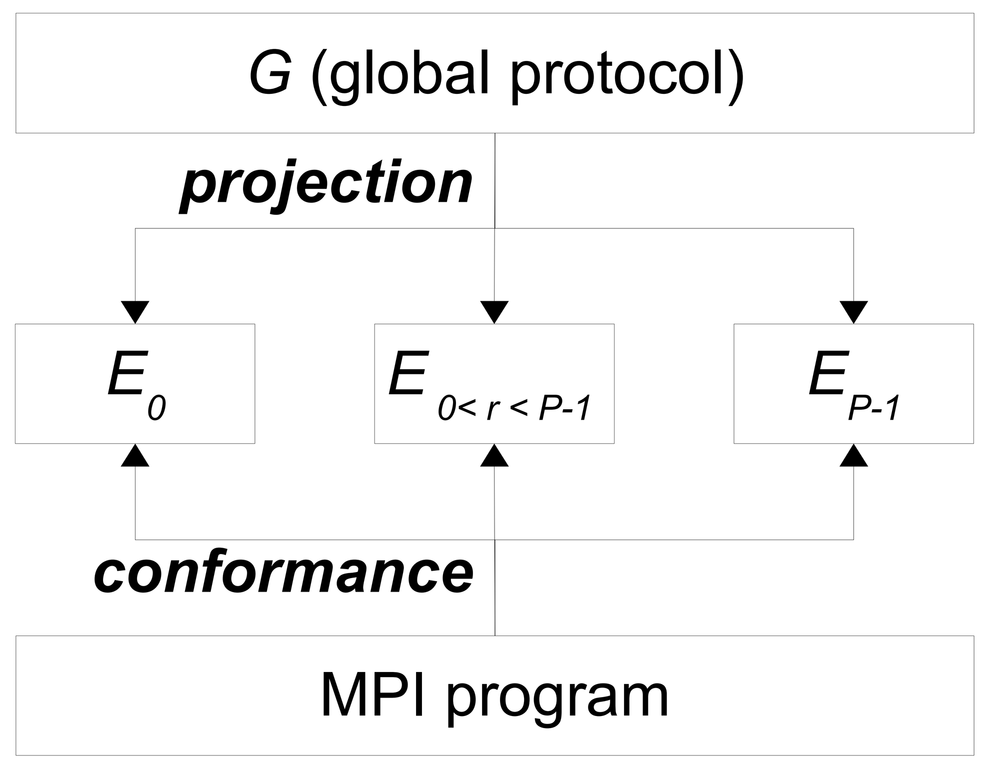
```
float err, localErr, out[N], in[N], ...;
int r, P;
MPI_Status status;
MPI_Comm_rank(MPI_COMM_WORLD, &r); // -> process rank
MPI_Comm_size(MPI_COMM_WORLD, &P); // -> number of processes
...
for (itr=0; itr < MAX_ITER && err > MAX_ERROR; itr++) {
 ...
 if (r < P-1) {
  // -> r+1 (right neighbor), executed for r = 0 ... P-2
  MPI_Send(out, N, MPI_FLOAT, r+1, 0, MPI_COMM_WORLD, &status);
 }

 if (r > 0) {
  // <- r-1 (left neighbor), executed for r = 1 ... P-1
  MPI_Recv(in, N, MPI_FLOAT, r-1, 0, MPI_COMM_WORLD, &status);
 }
 // some computation takes place and localErr is calculated
 ...
 // obtain global error (involves all processes)
 MPI_Allreduce(&localErr, &err, 1, MPI_FLOAT, MPI_SUM, MPI_COMM_WORLD);
}
```

KEY CHALLENGES

1. Refine multiparty session type abstractions to capture the general traits of MPI programs, e.g., rank-based communication, collective operations, and common communication patterns. Some particular features impose additional complexity, such as nondeterministic operations (e.g., wildcard receives).
2. Define and verify the conformance relation between MPI programs and multiparty session types. In essence, we need to determine a sound correspondence between a session type specification and the control structure of a MPI program for all its processes. This is **far from trivial**, as even the simple example above illustrates:
   - The communication flow is dependent on the process rank, i.e., for every participant `r` in the example an endpoint protocol must be found, matching the concrete control flow of the MPI process for rank `r`.
   - A control flow synchrony needs to be established between processes. In the example, we need to infer that all ranks execute the same number of loop iterations (as hinted by the **collective-loop** construct at the session type level), based on the assertion that `err` and `i` always have the same value in all processes per each iteration (note that `err` results from `MPI_Allreduce`).

SESSION TYPES



GLOBAL PROTOCOL
(for MPI program)

$G = \Pi\, P.\, \Pi\, N.$
foreach $(0 \leq r < P-1)$ {
  collective-loop {
    $r \longrightarrow r+1 \ \langle \texttt{float}, N \rangle$
    $\texttt{Allreduce} \ \langle \texttt{float}, N \rangle$
  }
}

ENDPOINT PROTOCOLS
(for the 3 groups of MPI processes)

$E_0 = \Pi\, N.$
collective-loop {
  $\longrightarrow 1 \ \langle \texttt{float}, N \rangle$
  $\texttt{Allreduce} \ \langle \texttt{float}, 1 \rangle$
}

$E_{0<r<P-1} = \Pi\, N.$
collective-loop {
  $\longleftarrow r-1 \ \langle \texttt{float}, N \rangle$
  $\longrightarrow r+1 \ \langle \texttt{float}, N \rangle$
  $\texttt{Allreduce} \ \langle \texttt{float}, 1 \rangle$
}

$E_{P-1} = \Pi\, N.$
collective-loop {
  $\longleftarrow P-2 \ \langle \texttt{float}, N \rangle$
  $\texttt{Allreduce} \ \langle \texttt{float}, 1 \rangle$
}

REFERENCES

[1] Gopalakrishnan, G., Kirby, R.M., Siegel, S., Thakur, R., Gropp, W., Lusk, E., De Supinski, B.R., Schulz, M., Bronevetsky, G.: Formal analysis of MPI-based parallel programs. Communications ACM 54(12), 82–91 (2011)

[2] Honda, K., Mukhamedov, A., Brown, G., Chen, T., Yoshida, N.: Scribbling interactions with a formal foundation. Distributed Computing and Internet Technology pp. 55–75 (2011)

[3] Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: POPL. pp. 273–284. ACM (2008)

[4] Siegel, S., Mironova, A., Avrunin, G., Clarke, L.: Combining symbolic execution with model checking to verify parallel numerical programs. ACM TOSEM 17(2), 1–34 (2008)

This work is supported by EPRSC funds EP/G015635/01 and EP/G015481/01, FCT funds PTDC/EIA-CCO/122547/2010 and CMU-PT/NGN/0044/2008, and the Ocean Observatories Initiative.