A Formal Specification of the MPI-2.0 Standard in TLA+

Guodong Li, Michael DeLisi, Ganesh Gopalakrishnan, Robert M. Kirby
School of Computing, University of Utah, Salt Lake City, UT 84112

The Problem
- Rigorous descriptions for widely used APIs essential
- Informal documents / Experiments not a substitute

Goals / Benefits
- Define MPI rigorously → Formal reasoning
- Permit running “Litmus Tests” → Understanding
- Tests for Implementations → Coverage
- Guide New Implementations → ‘what’ (not ‘how’)

The Methodology
- Formalize MPI-2.0 primitives in TLA+
- First Order Logic / Sets
- Use the TLC model checker to check properties
- Finitely Instantiate TLA+ Specs

C Front-end / VisualStudio Debugger
Error Trace Display thru Familiar GUI

Motivating Example
P0: if (rank==0){MPI_Irecv(rcvbuf1, from 1, req1);
MPI_Irecv(rcvbuf2, from 1, req2);
MPI_Wait(req1);
MPI_Wait(req2);...}
P1: if (rank==1){sendbuf1=6; sendbuf2=7;
MPI_Isend(sendbuf1, to 0);
MPI_Isend(sendbuf2, to 0);...}

The following simple questions can be answered:
1. Is it guaranteed that rcvbuf1 will eventually contain
   the message sent out of sendbuf1? (yes)
   ...; MPI_Wait(req2); Assert(rcvbuf1==6 ∧ rcvbuf2==7); ...

2. When can the buffers be accessed? (after
   MPI_Wait is called)
   ...; Assert(rcvbuf1==6 ∧ rcvbuf2==7); MPI_Wait(req1);...

3. Will the first receive always complete before the
   second? (No)
   ...; MPI_Wait(req2); MPI_Wait(req1); ...

Acknowledgments
This work was supported by
NSF award CNS-0509379 and Microsoft HPC Institutes

Verification Framework
- Insert a compilation phase in the Phoenix compiler.
- Automatically generate TLA+ model from C
- Verify the model using the TLC model checker.
- Replay an error trail in the Microsoft Visual Studio
  parallel debugger.

Sizes of major specification components :

<table>
<thead>
<tr>
<th>Main Module</th>
<th>#functions/lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point to Point Communication</td>
<td>35/800</td>
</tr>
<tr>
<td>Userdefined Datatype</td>
<td>27/550</td>
</tr>
<tr>
<td>Group &amp; Communicator Management</td>
<td>54/650</td>
</tr>
<tr>
<td>Inter Collective Communication</td>
<td>16/950</td>
</tr>
<tr>
<td>Topology</td>
<td>18/250</td>
</tr>
<tr>
<td>Environment Management in MPI 1.1</td>
<td>10/200</td>
</tr>
<tr>
<td>Process Management</td>
<td>10/250</td>
</tr>
<tr>
<td>One sided Communication</td>
<td>15/550</td>
</tr>
<tr>
<td>Inter Collective Communication</td>
<td>14/350</td>
</tr>
<tr>
<td>I/O</td>
<td>56/1000</td>
</tr>
<tr>
<td>Interface &amp; Environment in MPI 2.0</td>
<td>35/800</td>
</tr>
</tbody>
</table>

Example: (Synchronizing) Collective Communication
A loose synchronization protocol is implemented: in the first “enter”
phase, process proc will proceed to its next “wait” phase provided that it
hasn’t participated in the current synchronization (say syn) and syn’s
status is either “entering” or “vacant”. If all expected processes have
participated then syn’s status will advance to “leaving”. In the “leave”
phase, proc is blocked if syn is not in leaving status or proc has left. The
last leaving process will reset syn’s status to be “vacant”.

C Program: if (rank == 0) MPI_Bcast (&b,1, MPI_INT, 0, comm1)⇒

References
1. Guodong Li, Michael DeLisi, Ganesh Gopalakrishnan, and
   Robert M. Kirby, Formal specification of the MPI 2.0
   standard in TLA+, Principles and Practices of Parallel
   Programming (PPoPP), 2008.
2. Robert Palmer, Michael Delisi, Ganesh Gopalakrishnan,
   and Robert M. Kirby, An approach to formalization and
   analysis of message passing libraries, Formal Methods for
   Industry Critical Systems (FMICS) (Berlin), 2007, Best
   Paper Award

http://www.cs.utah.edu/formal_verification/mpitla/