P5

Learning Objectives:

- Write a protocol in BSPL, and identify safety/liveness issues
- Become familiar with Node-RED, a tool that can be used to wire together services
- Use hands-on protocol enactments using Node-RED to identify safety/liveness issues

Scenario

Consider the following scenario, loosely based on real-life medical practice for breast cancer diagnosis. (This scenario is based on a process described by a US Department of Health and Human Services Expert Panel.)

- A patient, Patricia, goes to a primary care physician, Primo.
- Primo examines the patient and possibly (if the case is suspicious) performs a core needle biopsy and sends the collected tissue specimen to a pathologist, Partho. In addition, Primo also sends Patricia to a radiologist, Radia.
- If Radia notices suspicious calcifications, she sends her findings to the Tumor Board. Otherwise, she sends a report to Primo.
- Partho analyzes the specimen, and performs ancillary studies. If he finds suspicious cells, he sends his findings to the Tumor Board. Otherwise, he sends a report to Primo.
- The Tumor Board is an organizational entity within a hospital that plays a distinct role. Its main function is to adjudicate disagreements between radiologists and pathologists. It informs Primo of its findings regarding Patricia. It also informs Radia and Partho of those findings so they can learn from them.
- Primo discusses the diagnosis with Patricia.

Tasks

See the sections on Protocheck and Node-RED tools to see how to use those tools to complete the tasks described here.

1. Produce a BSPL protocol that reflects the above scenario, taking into consideration the following points:
    - A protocol is concerned only with interactions, not with internal decision making.
    - A protocol is concerned with roles, not specific agents or people.
    - Model the patient's visits as an interaction (i.e., a message).
    - Make sure the relevant information flows to the various parties so they can act appropriately.
    - Introduce appropriate parameters, including keys.

- Notice that Primo receives findings from one or more of the other parties, and conveys those findings to Patricia. You should verify that your protocol avoids race conditions (see next item).
- A naive formulation of the protocol based on the scenario may fail safety or liveness.

Deliverable: A protocol constructed by you by hand although taking advantage of syntax verification provided by the Protocheck tool.

2. Use the Protocheck and Node-RED (see instructions below) to create a few enactments of the protocol and identify any safety or liveness issues. If you had identified such issues beforehand then you can use Node-RED to verify those enactments. Verify that the Protocheck tool also identifies the same safety and liveness issues.
   Deliverable: An explanation of any safety or liveness problems identified.
   A JSON flow file generated using the Node-RED tool along with instructions on creating enactments that fail safety or liveness.
3. Modify the protocol so that it avoids these problems, while capturing the essence of the problem scenario. Verify again using safety and liveness tests of Protocheck.
   Deliverable: A corrected protocol that satisfies all checks.
   A description of how the scenario described in this document will work with the modified protocol.

Protocol Tool

Use the following protocol checker tool to verify the syntax and the other properties of your protocol.

You can download the protocheck tool from github; directions for use are in the readme there.
https://gitlab.com/masr/protocheck
Note: Building one of the dependencies of protocheck, boolexpr, fails on Mac 10.14.6 (Mojave). However, it works on Ubuntu and is expected to work on Windows.

Specifically, you will find the following actions of the tool useful:

- syntax: for syntax check
- flow: to create flows for Node-RED from a given protocol
- safety: check for safety issues (to be used in task 3 of this project)
- liveness: check for liveness issues (to be used in task 3 of this project)

Tip: The safety and liveness check can potentially take several minutes (2 to 5 mins) to complete, depending on the protocol. These checks are CPU and Memory intensive and you should let one check terminate before starting another check.

Node-RED

Node-RED (https://nodered.org/) is a visual programming tool used to wire together hardware devices, APIs, and services primarily for IoT applications. For this project we will use this tool to enact the protocol by sending messages between different roles.

Install Node-RED:

Install Node-RED to run locally: https://nodered.org/docs/getting-started/local
If you are using a Debian (even if it is not Raspberry Pi) or RPM based Linux, you can use the scripts available at the link.

Node-RED tutorial:

Follow these short tutorials to get familiar with Node-RED: https://nodered.org/docs/tutorials/
Specifically, we will use the Inject and Debug nodes to enact protocol and observe messages.

Install BSPL nodes:

Install the following package to Node-RED using the instructions in the repository:
https://gitlab.com/masr/iot
Now you should be able to load the Node-RED flows generated by the Protocheck tool from the command line using:
$ node-red <flow-file>
And loading (or re-loading) http://localhost:/1880 in the browser.

Enact Protocol:

Make sure you are familiar with how to use Inject and Debug nodes from the Node-RED tutorial. You can enact the protocol by sending a sequence of messages by pressing the Inject button on the relevant messages. Observe the Debug log in the sidebar to see how bindings are created for different roles as a result of those messages.
The flow generated by Protocheck has some default values for the parameters in the Inject nodes. You can modify those if you want for different enactments. In addition, you can also give a name to individual Inject nodes to help easily identify the type of messages they send.
After enacting a protocol that presents safety/liveness issues, note the sequence of messages (sequence of Injects) to reproduce the issue. Save the flow by exporting it.

Deliverables (to be submitted)

1. A protocol constructed by you, based on the scenario described here, produced in task 1. You should only use the syntax verification feature provided by the Protocheck tool. Submit a plain text file with “before” in the filename, e.g., protocol_before.txt. (30 points)
2. A corrected protocol that satisfies all the checks, produced in task 3. Submit a plain text file with “after” in the filename, e.g., protocol_after.txt. (20 points)
3. One or more JSON flow files that can be used to recreate the enactments with safety/liveness issues identified in task 2. (10 points)
4. A pdf file with the following:
    a. An explanation of any safety or liveness problems identified by the tools in about 50-80 words each. (20 points)
    b. An explanation of how the corrected protocol handles the scenario provided in the problem. (10 points)
    c. Instructions on how to use the submitted JSON flow(s) to recreate safety/liveness issues. (10 points)

Help

Samuel Christie, a PhD student with Prof. Singh, who developed the Protocheck tool has kindly offered to help with this assignment.

Please use the message-board forum for this assignment to talk about general concerns and post questions about the tool and so forth. Please do not post any part of your answers publicly; instead, send questions specific to your approach via email. When you write to Samuel, cc Guoqiang and Professor Singh so we can try to help where possible. (As usual, we will try not to provide you solutions but will guide you where we can).