Overview

Atomicity is a correctness property specifically for *composite* protocols. That is, protocols made by composing together smaller modules.

Specifically, *atomicity* is the idea that protocol enactment should follow the *all-or-nothing* principle; either a protocol should be enacted completely, or it shouldn't even be started. A specification is incorrect if an enactment can leave some parts incomplete. Verifying that a protocol is atomic prevents agents from waiting indefinitely for their part to complete, and also helps highlight possible mistakes in the specification.

A protocol is atomic if:

1. All of its components are atomic in the context of the composition
2. If any component completes, the protocol must also be able to complete

Example 1: Purchase

The following protocol composition (Purchase) is *not* atomic:

```
Debit {
  roles B, S
  parameters in order key, out payment
  private acceptDebit

  S -> B: AcceptDebit[in order, out acceptDebit]
  B -> S: PayDebit[in order, in acceptDebit, out payment]
}

Credit {
  roles B, S
  parameters in order key, out payment
  private acceptCredit

  S -> B: AcceptCredit[in order, out acceptCredit]
  B -> S: PayCredit[in order, in acceptCredit, out payment]
}

Purchase {
  roles B, S // Buyer, Seller
  parameters out order key, out filled
  private payment
```

```
  B -> S: PlaceOrder[out order]
  Debit(B, S, order, payment)
  Credit(B, S, order, payment)
  S -> B: FillOrder[in order, in payment, out filled]
}
```

This example is nonatomic because the `Debit` and `Credit` protocols can both be initiated (there are no conflicts between the `AcceptCredit` and `AcceptDebit` messages, so both of them can be sent in the same enactment), but both of them *cannot* complete, since they both produce the `payment` parameter.

Scenario: Create Lab Order Workflow

There is a real and somewhat complex workflow for generic health lab work, covering collecting, analyzing, and reporting results, described here:
http://wiki.hl7.org/images/9/91/CreateLaboratoryOrderEventFlow.png

This diagram is a flowchart including a lot of unnecessary information, including private agent actions and state. Each “swim lane” between vertical lines represents the perspective of a different agent. The “POS” lanes are supposed to be the automated computer agents associated with each person, which for our purposes is an unnecessary distinction. Round bubbles are private actions or state, diamonds are choices, rectangles with points on the right are “send” actions, and corresponding rectangles with triangular dents on the left are “receive” actions. The arrow lines simply denote the sequence of events. Any lines across lanes must indicate some form of communication, so that the receiving party knows to continue the workflow.

Tasks

- Produce a corrected version of the above Purchase protocol, and explain why your corrected protocol is atomic. (**10pts**)
- Create a composite protocol capturing the Create Lab Order workflow (**20pts**)
    - Purely internal actions should not be represented
    - Use meaningful relationships between the data to guide the flow, rather than explicitly sequencing everything as in the flowchart
- Explain your protocol. Is your first attempt atomic? If not, make a second version that is, and explain what you had to change and why? (**20pts**)
- Design your own example protocol that illustrates an atomicity violation, and describe one way to resolve it. (**20pts**)
- [Bug bounty!] Find an example where the protocheck tool gives a false positive or negative for atomicity, or give some other useful suggestions for improving the tool. (**20pts**)

Note that the process protocheck uses for verifying protocols is *very* slow, particularly for larger protocols. If you want to be able to see more of what it's doing, and convince yourself it's not just hanging, run it with --verbose (-v), which will output intermediate statistics.

Deliverables (to be submitted)

1. A protocol illustrating an atomicity violation
2. A corrected version of the protocol
3. An explanation of both the violation and correction

Submit the protocols in separate plain text files, make sure they can be used as input files of protocheck tool. Submit your explanation in a pdf file.

Protocheck 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://github.com/shader/protocheck

Help

Samuel Christie <schrist@ncsu.edu> 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).