Musings on Protocols and Monads

ArmoredSoftware Team
palexand@ku.edu

May 19, 2015

Abstract

This document captures discussions on formally representing protocols using monadic constructs. This is a living document and will be updated frequently.

Notation

Throughout we use a trivial monadic notation for protocols that serves as our "assembly language" target for protocol compilation. The following conventions hold:

```
do {                % evaluate functions in sequence
     f(x);          % calculate f(x) and discard the result
     y <- f(x);     % calculate f(x) and bind the result to y
     send a $ x;    % evaluate x and send the result to a
     y <- receive a % receive data from a and the result to y
}
```

This is early work, so we play fast and loose with specific syntax and semantics. `send` and `receive` operate synchronously. Each `send` must have a corresponding `receive` to complete its operation.